Московский государственный университет имени М. В. Ломоносова

Факультет Вычислительной Математики и Кибернетики
Кафедра Математических Методов Прогнозирования

КУРСОВАЯ РАБОТА

«Применение методов регрессионного анализа для решения задачи прогнозирования временных финансовых рядов»

Выполнил:
студент 4 курса 417 группы
Остапец Андрей Александрович

Научный руководитель:
д.ф-м.н., доцент
Дьяконов Александр Геннадьевич

Москва, 2013

Содержание

1 Введение

Анализ и прогнозирование процессов, протекающих во времени, всегда были востребованными задачами. Задачи прогнозирования и поиска закономерностей во временных рядах возникают в различных сферах деятельности человека: медицине, экономике, физике, химии, метеорологии, кибернетике. С помощью временных рядов описываются процессы жизнедеятельности человека, стоимость акций на бирже, курсы валют, сигналы, погодные условия и т. д.

Для решения задач анализа временных рядов было предложено большое число методов. В том числе

- различные методы сглаживания и фильтрации (Р.Г. Браун, Ч. Хольт),
- авторегрессии и скользящего среднего (Дж. Бокс, Г. Дженкинс),
- модели, учитывающие гетероскедастичность (Р. Энгл).
- алгоритмы основанные на спектральных характеристиках (Д. Ваттс, Г. Дженкинс),
- статистические модели (Г. Крамер, А. Стюарт).
- были разработаны различные модели, представляющие закономерности в виде правил: ассоциативных (Р. Шрикант), эпизодических (Х. Манилла), иерархических (Ф. Морхен). Для поиска правил также создавались методы дискретизации временных рядов (Г. Дас).

2 Постановка задачи

Спрогнозировать цену финансового инструмента на основе обучающей выборки и параметров временных рядов предшествующего периода.

2.1 Исходные данные

Дано 2 файла разрешения «.csv», каждый из которых представляет собой таблицу с данными. В файле «Test» данные представлены в следующем виде:

	Период 1				Период 2				
Номер дня	Волатильность (в % годовых)	Угловой коэффициент линии регрессии	Смещение линии регрессии	Статистика R^2 (Коэффициент Детерминации)	Набл. 1	Набл. 2	...	Набл.80	Набл.102
1	v_1	k_1	b_1	r_1	p_1^1	p_1^2		p_1^{80}	p_1^{102}
...	...	...	...	...	...	...	...	...	...
...	v_{4191}	k_{4191}	b_{4191}	r_{4191}	p_{4191}^1	p_{4191}^2		p_{4191}^{80}	p_{4191}^{102}

Рис. 1: Формат данных

Все строки таблицы перемешаны, так чтобы информация в любой i-й строке была бесполезна для прогноза j-й строки. Параметры временного ряда в Периоде 1 построены по 500-м наблюдениям.

Наблюдение 1, Наблюдение 2, ..., Наблюдение 80, Наблюдение 102 - изменения котировок финансового инструмента в календарной последовательности, т.е. Наблюдение 1 - самые старые данные Наблюдение 80 - самые новые. Наблюдение 102 - это значение данного инструмента после 80+22 изменений с начала данных.

Второй файл «Control» содержит такой же вид столбцов данных, как и в предыдущем файле, только количество строк таблицы равно 2102 и без столбца «Наблюдение 102», который и нужно найти.

2.2 Оценка качества решения

Оценка модели производится двумя способами: через ошибку прогноза и экспертной оценкой специалистов АлгоМоста. Введем следующие обозначения:

$P(i)_L$ - последняя наблюдаемая цена, $P(i)_F$ - прогноз, $P(i)_R$ - значение цены, которое требуется спрогнозировать, $i = 1, \ldots, N$, где N - количество точек, которое требуется спрогнозировать.

Ошибка вычисляется следующим образом:

1. Если $P(i)_R < P(i)_L * (1 - \varepsilon)$, где $\varepsilon = 0.000088$, то

$$e(i) = \Big|\frac{P(i)_F}{P(i)_R} - 1\Big| + \Big(\frac{P(i)_F}{P(i)_L} - 1\Big)$$

2. Если $P(i)_L * (1 - \varepsilon) < P(i)_R < P(i)_L * (1 + \varepsilon)$, то

$$e(i) = \Big|\frac{P(i)_F}{P(i)_L} - 1\Big|$$

3. Если $P(i)_R > P(i)_L * (1 + \varepsilon)$ и $P(i)_F > P(i)_R$, то

$$e(i) = \Big|\frac{P(i)_F}{P(i)_R} - 1\Big| + \Big(\frac{P(i)_L}{P(i)_F} - 1\Big)$$

4. Если $P(i)_R > P(i)_L * (1 + \varepsilon)$ и $P(i)_F < P(i)_R$, то

$$e(i) = \Big|\frac{P(i)_F}{P(i)_R} - 1\Big| + \Big(\frac{P(i)_F}{P(i)_L} - 1\Big)$$

5. Итоговая ошибка считается следующим образом:

$$Error = \frac{100}{N}\sum_{i=1}^{N} e(i)$$

Можно отметить, что ошибка может принимать отрицательные значения.

3 Решение задачи регрессии

3.1 Анализ тестовых данных

Для начала рассмотрим, как выглядят тестовые данные. Построим гистограмму распределения изменения цены на 102 шаге относиельно цены на 80 шаге:

```
load("Test.RData") # загрузка обучающей выборки
norm_diff <- Test$price102/Test$price80 # нормированное изменение цены
hist(norm_diff,breaks=35) # построение гистограммы
```

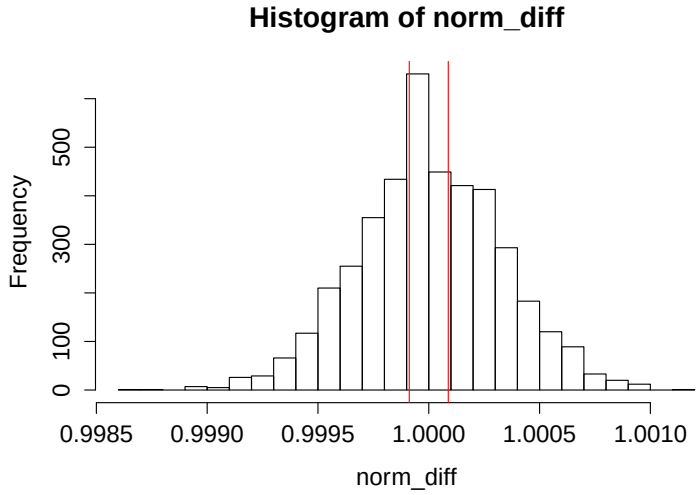

Рис. 2: Гистограмма распределения изменения цен

Красными линиями отмечены пороги $x = 1 - \varepsilon, x = 1 + \varepsilon$. Численное распределение значений относительно этих порогов:

```
> eps = 0.000088
> sum(norm_diff<=1-eps)
[1] 1662
> sum(norm_diff>=1+eps)
[1] 1710
> sum(norm_diff>1-eps & norm_diff<1+eps)
[1] 819
```

Обратим внимание на то, что не нужно предсказывать изменение цены на шагах с 81 по 101, надо предсказать лишь какой окажется цена на 102 шаге. На представленном ниже графике черным цветом показано изменение цены с 1 по 80 измерения, красной точкой отмечена цена на 102 шаге.

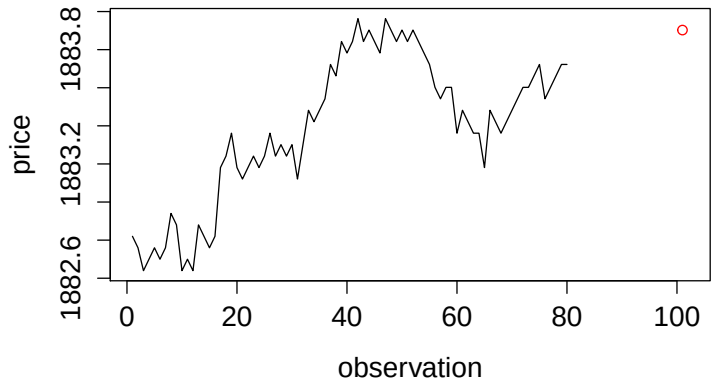

Рис. 3: Пример изменения цены

3.2 Эвристические решения

Для начала напишем функцию, которая будет оценивать полученные решения:

```
error <- function(F,R,L){
  err = 0
  for(i in 1:length(R))
  {
    if(R[i] <= L[i]*(1-eps))
      err = err + abs(F[i]/R[i]-1)+(F[i]/L[i]-1)
    else
      if(L[i]*(1-eps) < R[i] & R[i] < L[i]*(1+eps))
        err = err + abs(F[i]/L[i]-1)
      else
        if(R[i] >= L[i]*(1+eps) & F[i] > R[i])
          err = err + abs(F[i]/R[i]-1)+(L[i]/F[i]-1)
        else
          err = err + abs(F[i]/R[i]-1)-(F[i]/L[i]-1)
  }
  err*100/length(R)
}
```

Для начала посмотрим, какой результат получится, если подать этот функции в качестве предсказанных значений истинные:

```
> F = Test$price102
> R = Test$price102
> L = Test$price80
> error(F,R,L)
[1] -0.02462304
```

Теперь посмотрим, какой результат получится, если взять за предсказание последнюю наблюдаемую цену:

```
> F = Test$price80
> error(F,R,L)
[1] 0.02534652
```

Это первое решение и нужно попытаться улучшить этот результат.

Заметим, что если мы научимся предсказывать верно принадлежность изменение цены к одному из 3 классов (цена упала больше, чем на ε, осталось в пределах [-ε, ε], выросла больше, чем на ε), то получим следующий результат:

```
for(i in 1:length(R))
{
  if(R[i] <= L[i]*(1-eps))
    f[i] <- L[i]*(1-eps)
  else if(L[i]*(1-eps) < R[i] & R[i] < L[i]*(1+eps))
    f[i] <- L[i]
  else
    f[i] <- L[i]*(1+eps)
}

error(f,R,L)
[1] 0.01118594
```

3.3 Линейная регрессия

Попробуем оставлять несколько последних переменных и по ним строить линейную регрессию:

```
set.seed(1234) #задание начального состояния генератора случайных чисел
Info <- Test[,6:85] # оставляем только первые 80 наблюдений
M <- 10 # количество случайных генераций для усреднения ошибки
train_err <- rep(0,30) # вектор ошибок для обучения
test_err <- rep(0,30) # вектор ошибок для тестирования

for(j in 1:M)
{
  samp<-sample(1:4191,2800,replace=FALSE) # генерируем обучение
  for(i in 1:30) # настройка модели и подсчет полученной ошибки
  {
    dat = data.frame(obs = Info[samp,(81-i):80], T = R[samp])
    model <- lm(T ~ .,data = dat)
    test_err[i] <- test_err[i] +
                  error(predict(model,data.frame(obs=Info[-samp,(81-i):80])),R[-samp],L[-samp])
    train_err[i] <- train_err[i] +
                    error(predict(model,data.frame(obs=Info[samp,(81-i):80])),R[samp],L[samp])
  }
}
train_err <- train_err / M
test_err <- test_err / M
```

На графике красным цветом показана ошибка на обучении, зеленым цветом - ошибка на контроле.

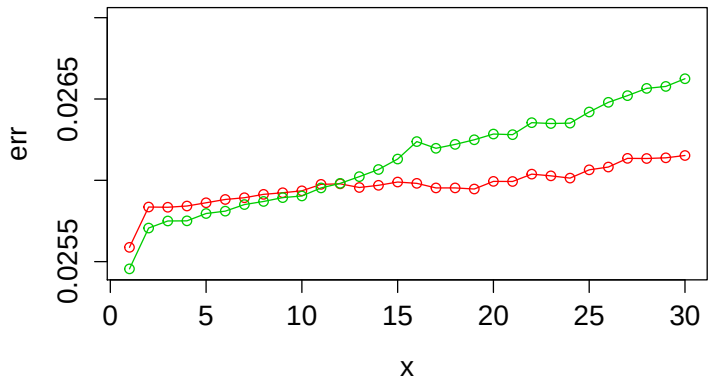

Рис. 4: График ошибок для линейной регрессии с обычными признаками

Теперь перейдем к новой матрице признаков, в которой признак - это частное от деления 2 соседних значений цен:

```
Test_norm <- Test[,6:85]
dimen <- dim(Test_norm)
Test_norm <- matrix(rep(rep(0,dimen[1]),dimen[2]),dimen[1],dimen[2])
for (j in 6:85)
{
  Test_norm[,j-5] <- Test[,j+1]/Test[,j]
}
```

После выполенения аналогичной процедуры получается следующий результат работы:

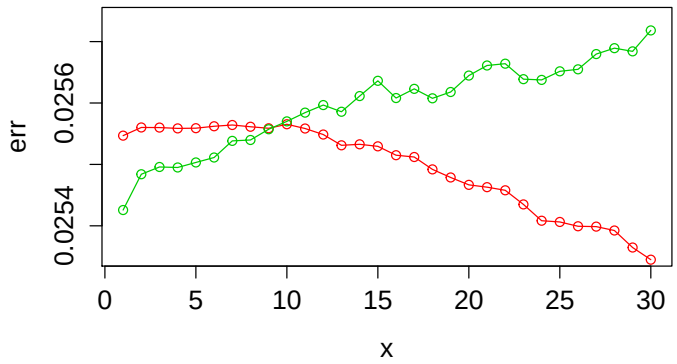

Рис. 5: График ошибок для линейной регрессии с нормированными признаками

Теперь рассмотрим новую матрицу признаков, в которой признак - это частное от деления 2 встречающихся значений цен:

```
Info <- Test[,6:85] # оставляем только первые 80 наблюдений
p <- 0
for (j in 80:i)
{
  for (k in 1:(i-1))
  {
    p <- p+1
    Test_norm_all[,p]<- Info[,j]/Info[,k]
  }
```

```
}
```

И вновь запускаем тестирование:

```
set.seed(1234) #задание начального состояния генератора случайных чисел
M <- 6 # количество случайных генераций для усреднения ошибки
train_err <- rep(0,15) # вектор ошибок для обучения
test_err <- rep(0,15) # вектор ошибок для тестирования
Info <- Test[,6:85] # оставляем только первые 80 наблюдений

for(j in 1:M)
{
  samp<-sample(1:4191,2800,replace=FALSE) # генерируем обучение
  for(i in 1:15) # настройка модели и подсчет полученной ошибки
  {
    p <- 0
    for (j in 80:(81-i))
    {
      for (k in 1:(80-i))
      {
        p <- p+1
        Test_norm_all[,p]<- Info[,j]/Info[,k]
      }
    }
    dat = data.frame(obs = Test_norm_all[samp,p], T = Test[samp,86]/Test[samp,85])
    model <- lm(T ~ .,data = dat)
    test_err[i] <- test_err[i] +
      error(predict(model,data.frame(obs=Test_norm_all[-samp,p]))*Test[-samp,85],R[-samp],L[-samp])
    train_err[i] <- train_err[i] +
      error(predict(model,data.frame(obs=Test_norm_all[samp,p]))*Test[samp,85],R[samp],L[samp])
  }
}
train_err <- train_err / M
test_err <- test_err / M
```

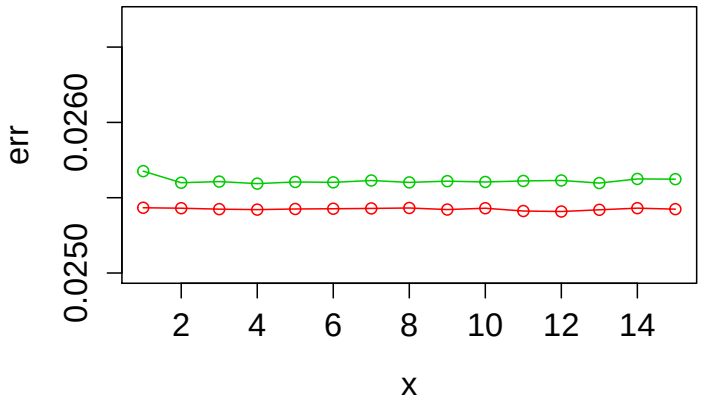

Рис. 6: График ошибок для линейной регрессии с 3 типом признаков

3.4 Random Forest

Для начала рассмотрим стандарнтные признаки и построим график качества решения от числа деревьев:

```
M <- 10 # количество случайных генераций для усреднения ошибки
```

```
train_err <- rep(0,10) # вектор ошибок для обучения
test_err <- rep(0,10) # вектор ошибок для тестирования

for(j in 1:M)
{
  samp<-sample(1:4191,2800,replace=FALSE) # генерируем обучение
  p <- 0
  for(i in seq(100,1000,100)) # настройка модели и подсчет полученной ошибки
  {
    p <- p + 1
    rf <- randomForest(Test[samp,2:85],Test[samp,86],ntree = i)
    test_err[p] <- test_err[p] +
                  error(predict(rf,Test[-samp,2:85]),R[-samp],L[-samp])
    train_err[p] <- train_err[p] +
                    error(predict(rf,Test[samp,2:85]),R[samp],L[samp])
  }
}
train_err <- train_err / M
test_err <- test_err / M
```

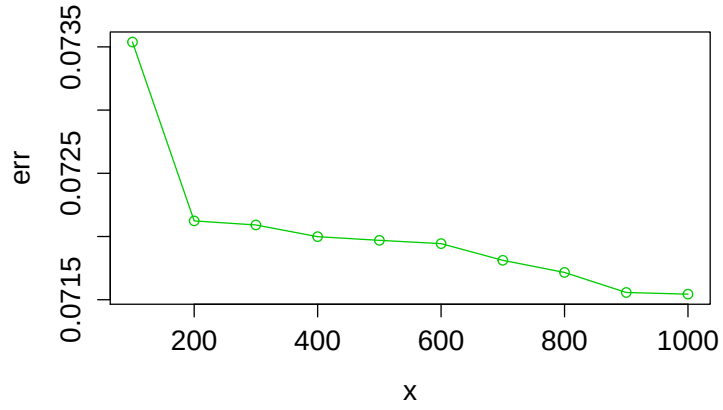

Рис. 7: График качества на контрольной выборке

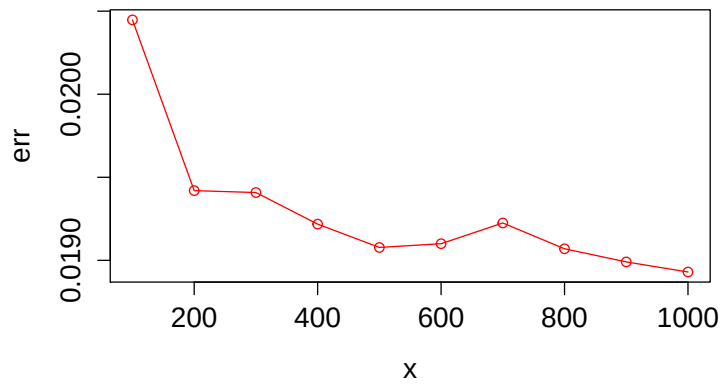

Рис. 8: График качества на обучении

Видно, что происходит сильное переобучение. Теперь построим график зависимости ошибки от mtry - он определяет, сколько столбцов тестируется при поиске следующего разбиения, для каждого

дерева.

```
.......................
for(i in seq(5,75,10))
.......................
rf <- randomForest(Test[samp,2:85],Test[samp,86],ntree = 200, mtry = i)
.......................
```

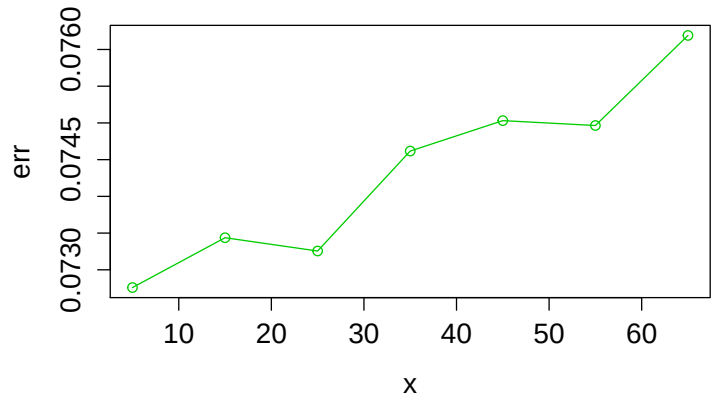

Рис. 9: График качества на контрольной выборке от параметра mtry

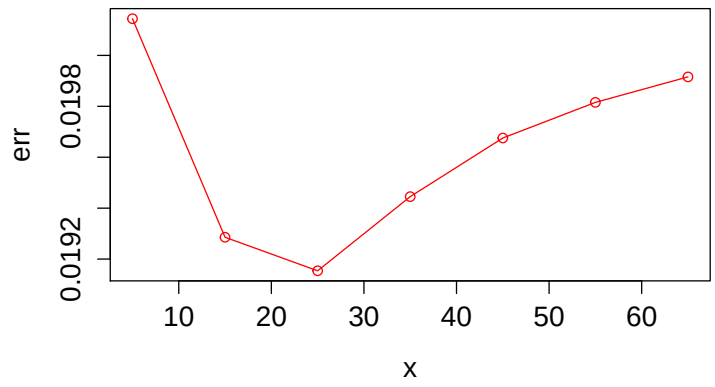

Рис. 10: График качества на обучении от параметра mtry

Теперь рассмотрим поведение графика для второго типа признаков - первые 5 из исходных + частные от произведения сосседних цен.

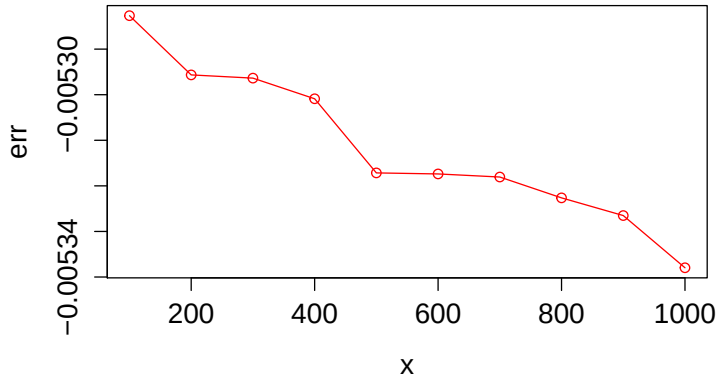

Рис. 11: График качества на контрольной выборке

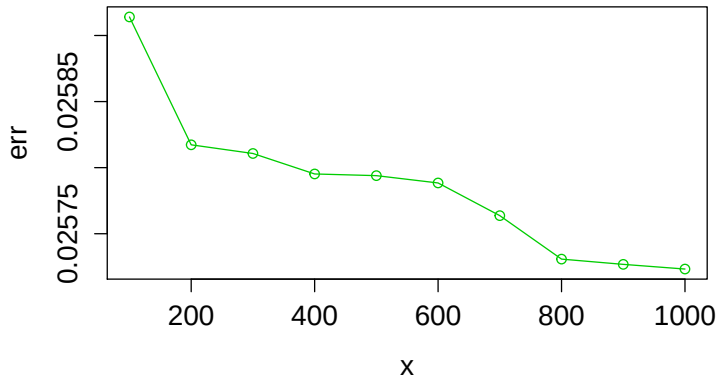

Рис. 12: График качества на обучении

Аналогично, построим график качества от параметра mtry:

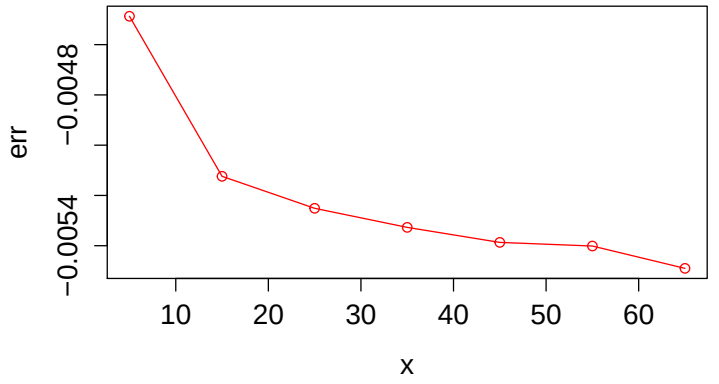

Рис. 13: График качества на контрольной выборке от параметра mtry

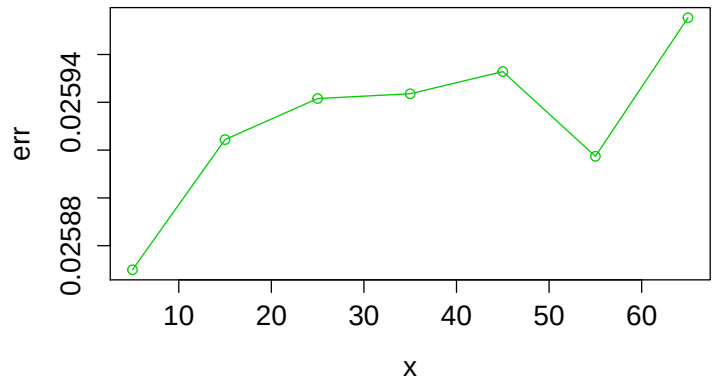

Рис. 14: График качества на обучении от параметра mtry

3.5 GBM

Матрица, где в качестве признаков взяты попарные отношения соседних цен:

```
data.train = data.frame(X = Test_norm[samp,pr], Y = Test_norm[samp,85])
gbm1 <- gbm(Y ~ ., data.train , distribution = "laplace",
            n.trees = 10000,bag.fraction = 0.5,
            train.fraction = 0.9)
```

Черным показано ошибка на обучении, красным - ошибка на контроле. По оси абсцисс отложено число деревьев.

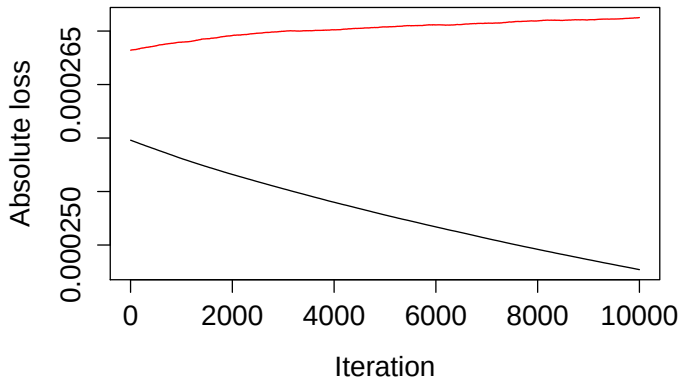

Рис. 15: График качества от числа деревьев

Но на независимой тестовой выборке качество равно:

```
[1] 0.02537242
```

Изменим ошибку:

```
gbm(Y ~ ., data.train , distribution = "gaussian",....)
```

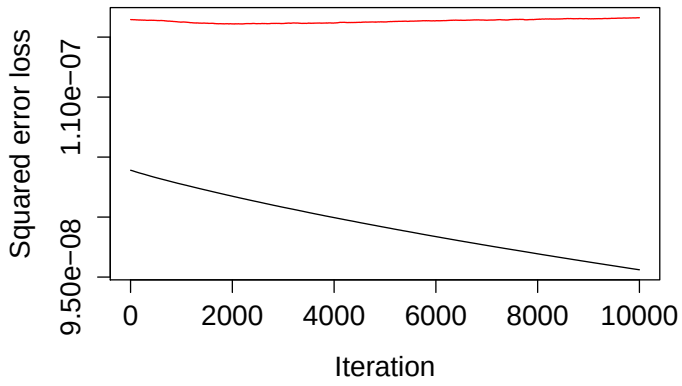

Рис. 16: График качества от числа деревьев

```
[1] 0.02645909
```

Теперь рассмотрим матрицу, где в качестве признаков взяты попарные разности соседних значений цен:

```
data.train = data.frame(X = Test_razn[samp,pr], Y = Test_razn[samp,81])
gbm(Y ~ ., data.train , distribution = "gaussian",....)
```

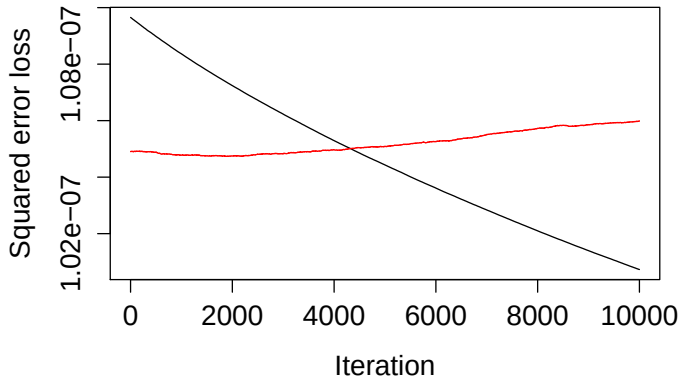

Рис. 17: График качества от числа деревьев

В итоге, средняя ошибка при тестировании на контроле составила:

```
0.02536004
```

Лучшее количество деревьев равно примерно 190.

Изменим ошибку:

```
gbm(Y ~ ., data.train , distribution = "laplace",....)
```

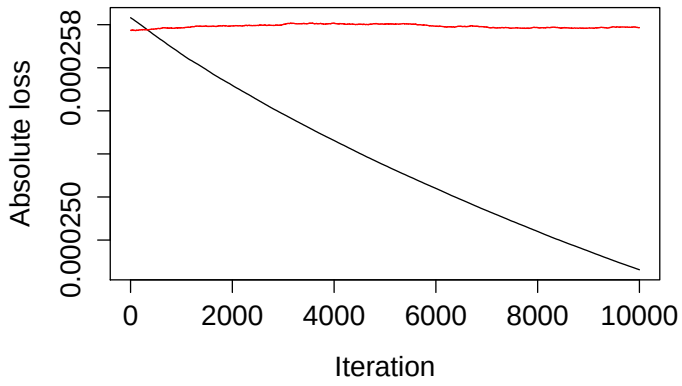

Рис. 18: График качества от числа деревьев

```
[1] 0.02533281
```

3.6 Настройка линейной комбинации

```
te<-0 # ошибка на контроле
tre<-0 # ошибка на обучении
for (i in 1:15){ # число запусков
  samp<-sample(1:4191,2800,replace=FALSE) # генерируем обучение
  pr <- 1:80

  # GBM
  data.train = data.frame(X = Test_razn[samp,pr], Y = Test_razn[samp,81])
  gbm1 <- gbm(Y~., data.train , distribution = "gaussian",
              n.trees = 500,bag.fraction = 0.5,
              train.fraction = 0.9)

  best.iter <- gbm.perf(gbm1,method="OOB")
  test1[,1] <- predict.gbm(gbm1,data.frame(X = Test_razn[-samp,pr]), best.iter)
  tr1[,1] <- predict.gbm(gbm1,data.frame(X = Test_razn[samp,pr]), best.iter)

  # RF
  rf <- randomForest(Test_razn[samp,pr],Test_razn[samp,81],ntree = 800, mtry = 40)

  test1[,2]<-predict(rf,Test_razn[-samp,1:80])
  tr1[,2]<-predict(rf,Test_razn[samp,1:80])

  # последнее изменение цены
  test1[,3]<- Test_razn[-samp,80]
  tr1[,3]<- Test_razn[samp,80]

  # Настройка линейной комбинации
  dat<- data.frame( F = tr1, T = Test_razn[samp,81])
  l <- lm(T~.,data = dat)
  te<- te + error(predict(l,data.frame(F = test1))+Test[-samp,85],R[-samp],L[-samp])
  tre<- tre + error(predict(l,data.frame( F = tr1))+Test[samp,85],R[samp],L[samp])
}
```

Получились следующие результаты:

```
> te/15
[1] 0.02543287
> tre/15
[1] 0.02527602
```

4 SSA

SSA (Singular spectrum analysis) - метод анализа временных рядов, основанный на преобразовании одномерного временного ряда в многомерный ряд с последующим применением к полученному многомерному временному ряду метода главных компонент.

Способ преобразования одномерного ряда в многомерный представляет собой «свертку» временного ряда в матрицу, содержащую фрагменты временного ряда, полученные с некоторым сдвигом. Общий вид сдвиговой процедуры напоминает «гусеницу», поэтому сам метод нередко так и называют - «Гусеница»: длина фрагмента называется длиной «гусеницы», а величина сдвига одного фрагмента относительно другого шагом «гусеницы».

```
library("Rssa")
e <- 0
  samp<-sample(1:4191,500,replace=FALSE)
for (j in 1:10) # на какую длину идет предсказание
{
  f <- 1
  for(i in 1:500){
    s <- ssa(Test[samp[i],6:85],kind = "1d-ssa")
    f[i] <- forecast(s, groups = list(1:3),
                     len = j)$mean[j] # в качестве ответа выдаем последнее предсказанное значение
  }
  e[j] <- error(f,Test[samp,86],Test[samp,85])
}
```

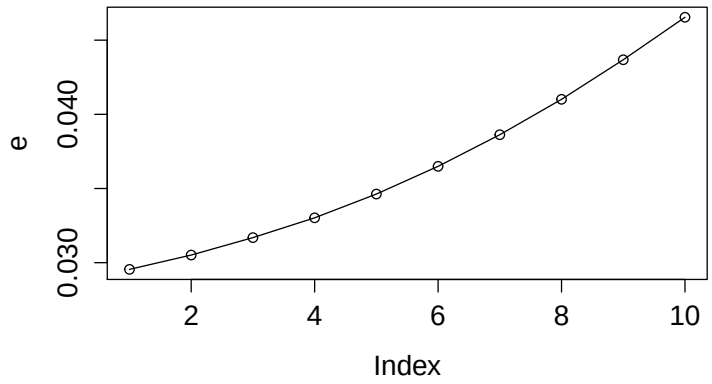

Рис. 19: График качества от длины предсказания

Поменяем тип предсказания:

```
.....................................
s <- ssa(Test[samp[i],6:85],kind = "toeplitz-ssa")
.....................................
```

Полученный результат:

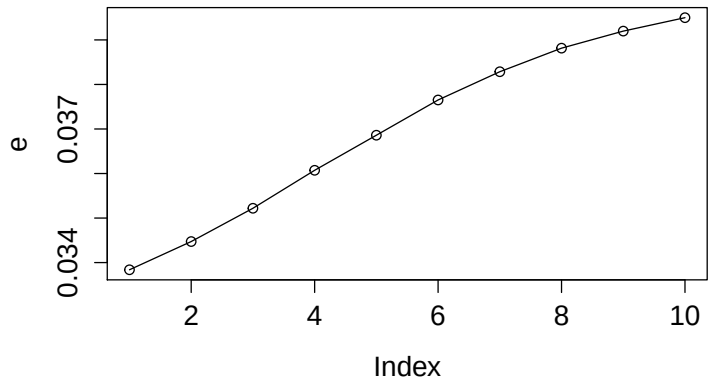

Рис. 20: График качества от длины предсказания

Попробуем теперь предсказывать на одну точку вперед, но рассмотрим различные длины временного ряда по которому строится предсказание:

```
.....................................
for (j in 4:20)
.....................................
s <- ssa(Test[samp[i],6:85],L = j)
.....................................
```

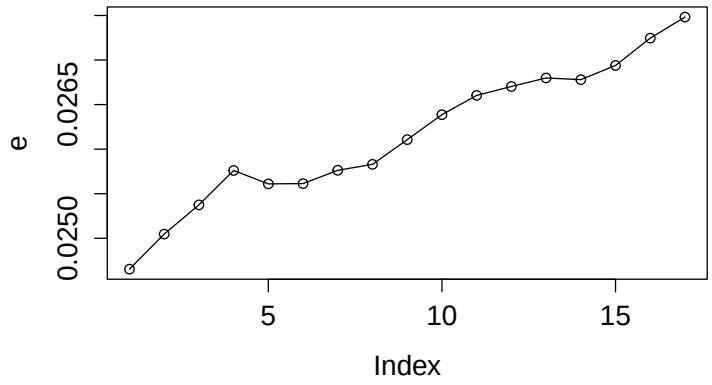

Рис. 21: График качества от последовательности, по которой строится предсказание

Лучший достигнутый результат:

```
> e[1]
[1] 0.02465364
```

5 Решение задачи классификации

Данную задачу можно решать и как задачу классификации. Для этого смотрелось отклонение цены на 102 шаге относительно цены на 80 шаге и на основе этого присуждался номер класса. Всего было выбрано 8 классов.

```
for(i in 1:length(R))
{
  if(R[i] <= L[i]*(1-3*eps))
```

```
    tkm[i]<-0
  else if(R[i] <= L[i]*(1-2*eps))
    tkm[i]<-1
  else if(R[i] <= L[i]*(1-1*eps))
    tkm[i]<-2
  else if(R[i] <= L[i]*(1-0*eps))
    tkm[i]<-3
  else if(R[i] <= L[i]*(1+1*eps))
    tkm[i]<-4
  else if(R[i] <= L[i]*(1+2*eps))
    tkm[i]<-5
  else if(R[i] <= L[i]*(1+3*eps))
    tkm[i]<-6
  else
    tkm[i]<-7
}
```

Попробуем теперь настроить kNN. Для этого рассмотрим оценку качества при различном числе соседей.

```
e<-rep(0,100)
samp<-sample(1:4191,2800,replace=FALSE) # генерируем обучение
for (i in 1:100){
  an<-knn(Test_norm[samp,6:84],Test_norm[-samp,6:84],tkm[samp], k = i)
  an1 <- rep(0,length(R[-samp]))
  an<-as.numeric(an)
  for(j in 1:length(R[-samp])){
    if(an[j] <= 2 )
      an1[j] <- -eps
    else if(an[j] > 4)
      an1[j] <- eps
    else
      an1[j] <-0
  }
  f<- Test[-samp,85]*(1 +an1)
  e[i]<-error(f,R[-samp],L[-samp])
}
```

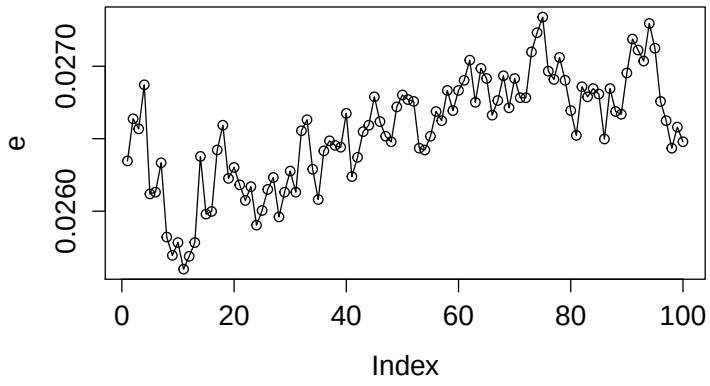

Рис. 22: График качества решения от количества соседей

6 Заключение

В ходе работы было показано, что задача прогнозирования изменения цены через достаточно большое количество измерений является трудной задачей. Все исследуемые алгоритмы, в лучшем случае, показали качество соизмеримое с качеством самого простого алгоритма - выдачей в качества ответа последней наблюдаемой цены.

Список литературы

[1] Дж.Бокс, Г.Дженкинс. Анализ временных рядов. Прогноз и управление. - М.: Мир, 1974

[2] Главные компоненты временных рядов: метод «Гусеница». Сб. статей. Ред. Д.Л.Данилов и А.А.Жиглявский. - СПбГУ, 1997

[3] P.H.Franses. Time series models for business and economic forecasting. - Cambridge Univ. Press, 1998.

[4] T.C.Mills. The econometric modeling of financial time series. - Cambridge Univ. Press, 1999