САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ – ПРОЦЕССОВ УПРАВЛЕНИЯ

КАФЕДРА ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ

Мамедли Икрам Илхам оглы

Дипломная работа

Разработка приложения для прогнозирования затрат и выручки предприятия общественного питания

Заведующий кафедрой,
кандидат физ.-мат. наук,
доцент Сергеев С. Л.

Научный руководитель,
кандидат физ.-мат. наук,
доцент Должиков В. В.

Рецензент Митрофанов Е. П.

Санкт-Петербург

2016

Введение

В настоящей работе рассмотрена деятельность предприятия общественного питания. Предприятие представляет собой кафе, в меню которого входят все типы блюд (первое, второе, салаты, десерты и пр.). По сути, предприятие является системой массового обслуживания (СМО); для математического моделирования его работы применимы выводы теории массового обслуживания, в частности ее основателя А.К. Эрланга.

Целью данной работы является повышение эффективности работы предприятия, в частности оптимизация расходов и трудозатрат персонала. К примеру, исходя из спроса на присутствующие в меню блюда в течение определённого периода, можно сделать прогноз на количество продуктов, требующих закупки в ближайшее время. Кроме того, проведя исследование изменения спроса на блюда определённого типа в течение суток, можно скорректировать рабочие часы персонала, занятого приготовлением блюд на кухне.

Поставленная задача оптимизации расходов и трудозатрат персонала разбивается на следующие подзадачи:

1) накопление и структурирование информации о заказах, сделанных посетителями кафе;
2) получение данных об общем количестве различных продуктов, потребовавшихся для приготовления блюд на заданном временном интервале;
3) прогнозирование количества заказов и выручки в краткосрочной перспективе;
4) предоставление данных для анализа занятости персонала, ответственного за приготовление того или иного типа блюд.

Постановка задачи

Задача состоит в разработке программного приложения, которое позволяет работать с накопленными данными, анализировать их и производить прогнозирование.

Для решения задачи предприятием общественного питания были подготовлены данные о количестве и времени заказов, сделанных посетителями за отчётный период с 1 февраля 2016 года по 1 апреля 2016 года.

Данные представляют собой список записей вида «дата – время – наименование блюда – количество порций», сформированных в виде таблицы в файле .csv. Программа, реализованная в среде программирования Qt, должна произвести разбор этого файла и сформировать соответствующую таблицу данных.

С помощью отдельно загруженной информации об ингредиентах, необходимых для приготовления каждого из блюд, присутствующих в ассортименте, программа должна сформировать суммарную информацию о количестве каждого вида продуктов, использованных в течение отчётного периода.

Графический интерфейс программы должен позволять выводить информацию об общем количестве блюд, требовавших приготовления в зависимости от времени суток. Эта информация может быть использована для анализа занятости поваров, специализирующихся на отдельных типах блюд (супы, горячее, салаты и др.), а также количества официантов, необходимых для обслуживания зала.

Модуль прогнозирования должен позволять получать необходимую информацию о предполагаемом значении временного ряда (ряда заказов) на следующий отчетный период по имеющимся статистическим данным.

Для построения прогноза были рассмотрены сначала примитивные

модели экспоненциального сглаживания (главным образом, модель Хольта-Уинерса), однако в работе приведена реализация более сложной и точной модели ARIMA (модели Бокса-Дженкинса), являющейся интегрированной композицией метода авторегрессии (AR) и модели скользящего среднего (MA). Модель реализована средствами программного продукта Matlab 2013a.

Обзор литературы

На первом этапе работы были использованы примитивные методы прогнозирования, в основании которых заложена рекуррентная модель экспоненциального сглаживания. Подробное описание моделей данного типа, а также частные случаи моделей экспоненциального сглаживания: модель с аддитивным сезонным эффектом (Тейлор-Вейдж) и мультипликативным сезонным эффектом (Уинтерс) подробно рассмотрены в учебном пособии Ю.П. Лукашина «Адаптивные методы краткосрочного прогнозирования» [1], где также представлен общий вид модели (обобщенная модель Брауна). Среди прочего в данном пособии отмечено, что все девять моделей экспоненциального сглаживания, вне зависимости от выбора характера уровня роста и характера сезонного эффекта, резко реагируют на импульс (разовое явление, имеющее место лишь в данный момент времени) ввиду постоянных параметров адаптации. Это объясняет нецелесообразность использования примитивных моделей для прогнозирования для рассмотренных в настоящей работе временных рядов большим количеством непрогнозируемых импульсов, связанных, например, праздничными и выходными датами.

Основу для работы составил метод Бокса-Дженкинса, введенный в использование в книге этих авторов [2]. Примечательно, что в данном издании основное внимание уделено практическому применению моделей авторегрессии, в особенности программированию на ЭВМ, несмотря на то, что исторически впервые полное описание модели появилось именно в этом труде.

Также для выбора параметров применяемой модели ARIMA были использованы информационный критерий Акаике [3] и байесовский критерий Шварца [1].

Также для представления о математическом моделировании систем

массового обслуживания с ожиданием был рассмотрен ряд источников в данной предметной области: пособия по теории массового обслуживания [4], [5].

Глава 1. Модели авторегрессии – скользящего среднего

1.1. Авторегрессионная модель AR

В авторегрессионной модели текущее значение процесса выражается через конечную линейную совокупность предыдущих значений процесса и возмущения (белого шума) [1]. Авторегрессионный процесс порядка p (AR(p)-процесс) определяется следующим образом:

$$X_t = c + \sum_{i=1}^{p} a_i X_{t-i} + \varepsilon_t,$$

где a_i – параметры модели (коэффициенты авторегрессии), c – постоянная (часто для упрощения предполагается равной нулю), а ε_t – белый шум.

Если ввести лаговый оператор $L : Lx_t = x_{t-1}$, то авторегрессионную модель можно представить следующим образом:

$$X_t = c + \sum_{i=1}^{p} a_i L^i X_t + \varepsilon_t.$$

В этой модели p+2 неизвестных параметра $c, a_1, a_2, ..., a_p, \sigma^2$, которые должны быть оценены по имеющимся данным об изучаемом процессе. Для решения практических задач, как правило, достаточно $p \le 2$.

1.2. Модель скользящего среднего MA

Модель скользящего среднего q-го порядка (MA(q)) – модель временного ряда вида:

$$X_t = \sum_{i=0}^{q} b_i \varepsilon_{t-j},$$

где ε_t – белый шум, b_i – параметры модели (b_0 можно считать равным 1 без ограничения общности) [1].

С помощью лагового оператора данную модель можно записать следующим образом:

$$X_t = (1 + \sum_{i=1}^{q} b_i L^i)\varepsilon_t$$

Она содержит q + 2 неизвестных параметра $c, b_1, b_2, ..., b_p, \sigma^2$. Обычно q = 0, 1, 2.

1.3. Модель авторегрессии – скользящего среднего AR(I)MA

Для достижения большей гибкости при построении модели исследуемых процессов полезно включать в нее и члены скользящего среднего, и авторегрессионные члены. Это приводит к смешанной модели ARMA(p,q) [1]:

$$X_t = c + \varepsilon_t + \sum_{i=1}^{p} a_i L^i X_t + (1 + \sum_{i=1}^{q} b_i L^i)\varepsilon_t$$

Эта модель содержит p+q+2 неизвестных.

Указанные выше модели считают, что временной ряд стационарен, то есть, его свойства не меняются во времени. На практике это часто бывает не так, и при наличии линейного роста модели оказываются неэффективны. Следующая модификация модели, ARIMA(p,d,q) (модель Бокса-Дженкинса) позволяет с помощью лагового оператора привести ряд к стационарному виду. Для этого производится взятие разностей порядка d от исходного временного ряда. Модель ARIMA(p,d,q) означает, что разности временного ряда порядка d подчиняются модели ARMA(p,q):

$$(1 - L)^d X_t = c + \varepsilon_t + (\sum_{i=1}^{p} a_i L^i)(1 - L)^d X_t + (1 + \sum_{i=1}^{q} b_i L^i)\varepsilon_t$$

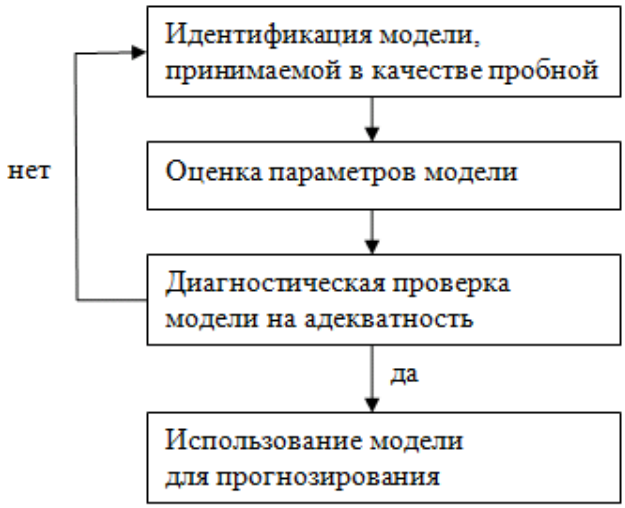


Рисунок 1. Общая схема выбора модели.

Алгоритм выбора модели ARIMA(p,d,q):

1. Вычисляем значение автоковариации в зависимости от лага для данного ряда (d=0) и разностных рядов первого и второго порядка (d=1,2) и выбираем такой порядок d, при котором величина автоковариации резко уменьшается при небольшом лаге.

2. С помощью критериев Акаике (AIC) и байесовского критерия Шварца (SBC) выбираем наиболее подходящие значения p и q.

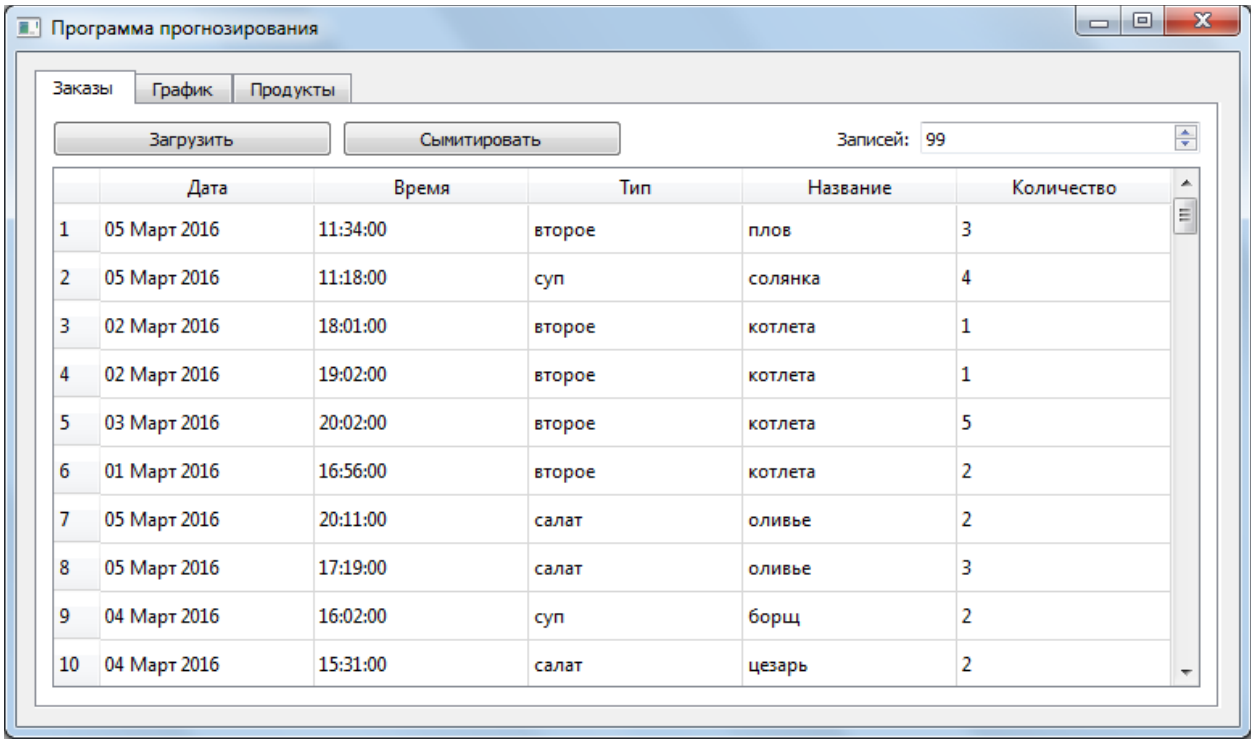
3. Оцениваем параметры моделей для данного ряда при выбранных значениях p,d,q методов наименьших квадратов.

4. Производим прогноз на выбранном горизонте прогнозирования с использованием выбранных параметров модели.

Глава 2. Программная реализация

Основная часть программы и графический интерфейс реализованы в среде Qt Creator на языке программирования C++. Модуль для выбора модели AR(I)MA и вычисления прогноза реализован в среде Matlab 2013a.

Графический интерфейс представляет собой окно с тремя вкладками: «Заказы», «График» и «Продукты».



	Дата	Время	Тип	Название	Количество
1	05 Март 2016	11:34:00	второе	плов	3
2	05 Март 2016	11:18:00	суп	солянка	4
3	02 Март 2016	18:01:00	второе	котлета	1
4	02 Март 2016	19:02:00	второе	котлета	1
5	03 Март 2016	20:02:00	второе	котлета	5
6	01 Март 2016	16:56:00	второе	котлета	2
7	05 Март 2016	20:11:00	салат	оливье	2
8	05 Март 2016	17:19:00	салат	оливье	3
9	04 Март 2016	16:02:00	суп	борщ	2
10	04 Март 2016	15:31:00	салат	цезарь	2

Рисунок 2. Вкладка «Заказы»

С помощью кнопки «Загрузить» пользователь загружает файл с таблицей данных о заказах посетителей. При этом формируется табличная модель, отображение которой производится во вкладке «Заказы» с помощью технологии Model/View. Сама модель представляет собой класс, отнаследованный от стандартного класса QAbstractTableModel.

Вкладка «График» позволяет построить график количества заказов за выбранный промежуток времени. Имеется возможность группировать блюда по их типу, чтобы оценить занятость соответствующего персонала на кухне. Виджет использует библиотеку для построения графиков QCustomPlot [6].

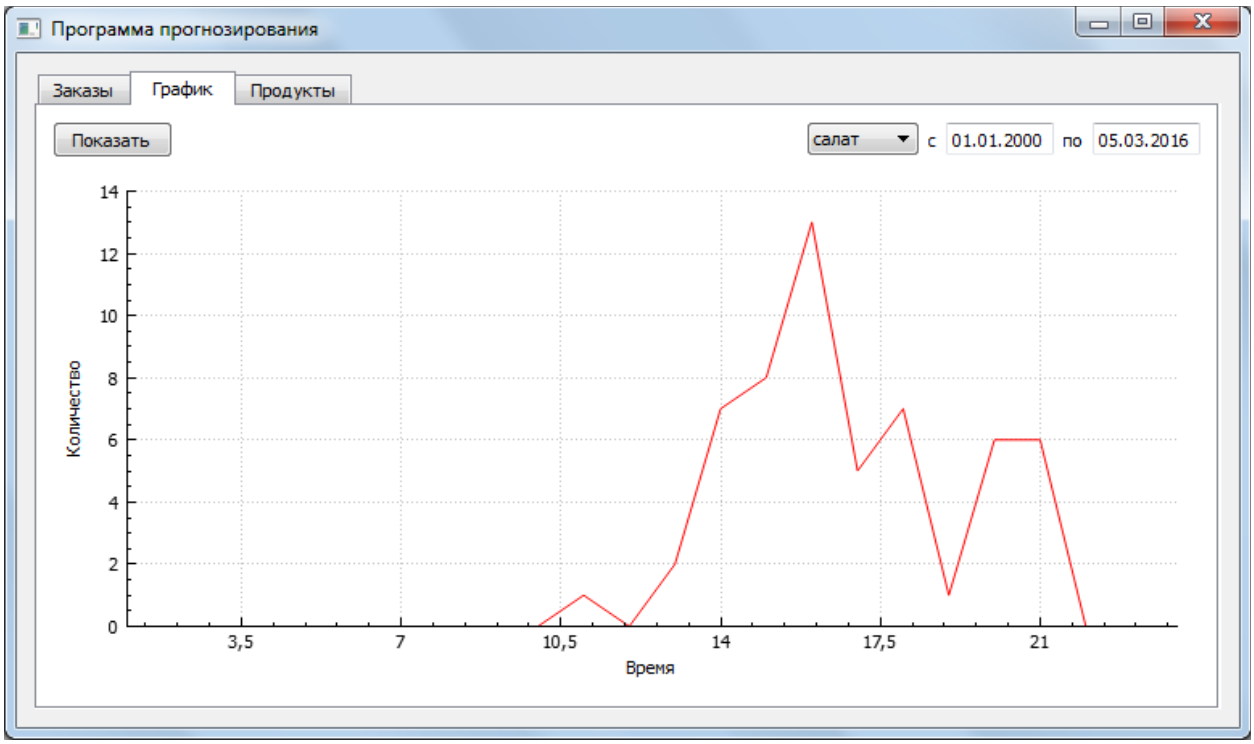


Рисунок 3. Вкладка «График»

Вкладка «Продукты» позволяет получить информацию о продуктах и их количестве, потраченных на приготовление блюд за выбранный промежуток времени. Кнопка «Прогноз» производит прогнозирование этого количества на следующую неделю.

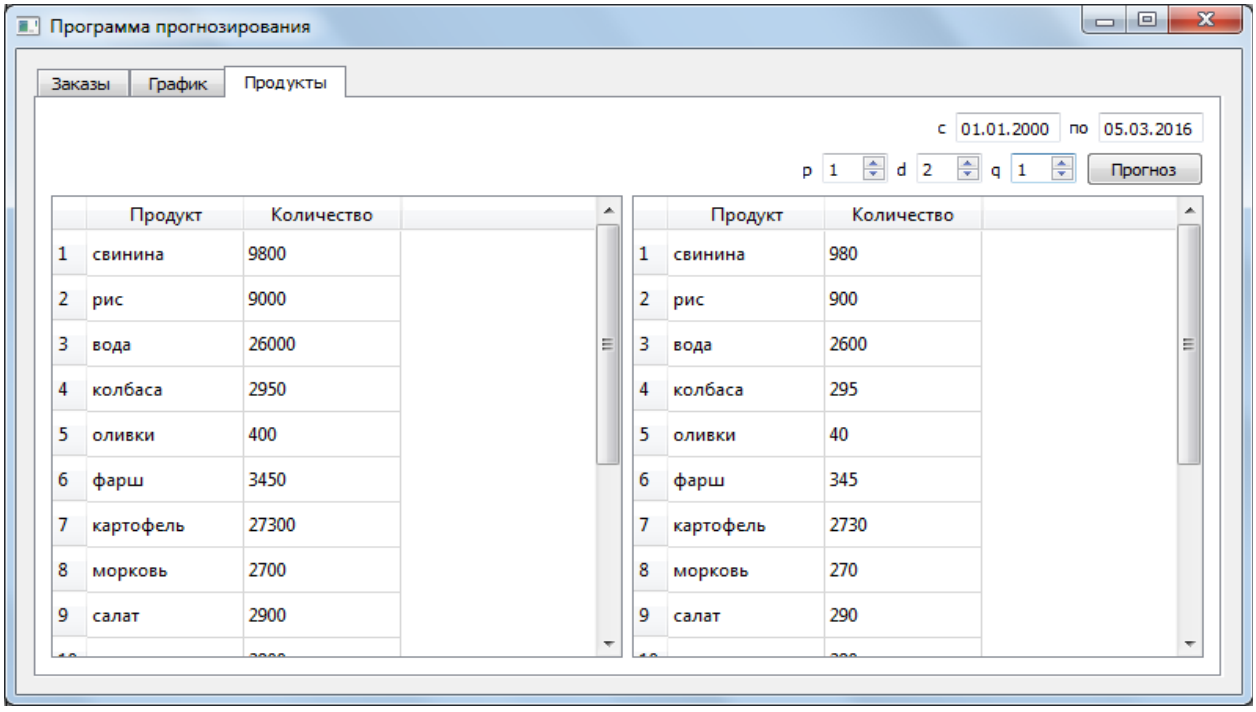


Рисунок 4. Вкладка «Продукты»

В работе программы используются классы Ingr (ингредиент), Dish (блюдо), DishList (меню), Order (заказ), IngrModel (модель для отображения списка ингредиентов) и Cuisine (список заказов и модель для их отображения).

Классы Cuisine и DishList реализованы с помощью шаблона проектирования «Singleton», создающего ровно один экземпляр этого класса instance* со статическим методом доступа getInstance().

Математическая часть программы использует автономное приложение, созданное из скриптового файла на языке Matlab, запускающееся с помощью библиотек Matlab Runtime Compiler. Это позволяет пользоваться функцией прогнозирования без специальной установки среды программирования Matlab.

Выводы

На рисунке 5 приведён график заказов супов. По нему можно сделать наблюдение о том, что пик заказов приходится на промежуток с 12 часов дня до 17 часов. С помощью этой информации был сделан вывод об оптимальном времени продажи комплексных обедов.

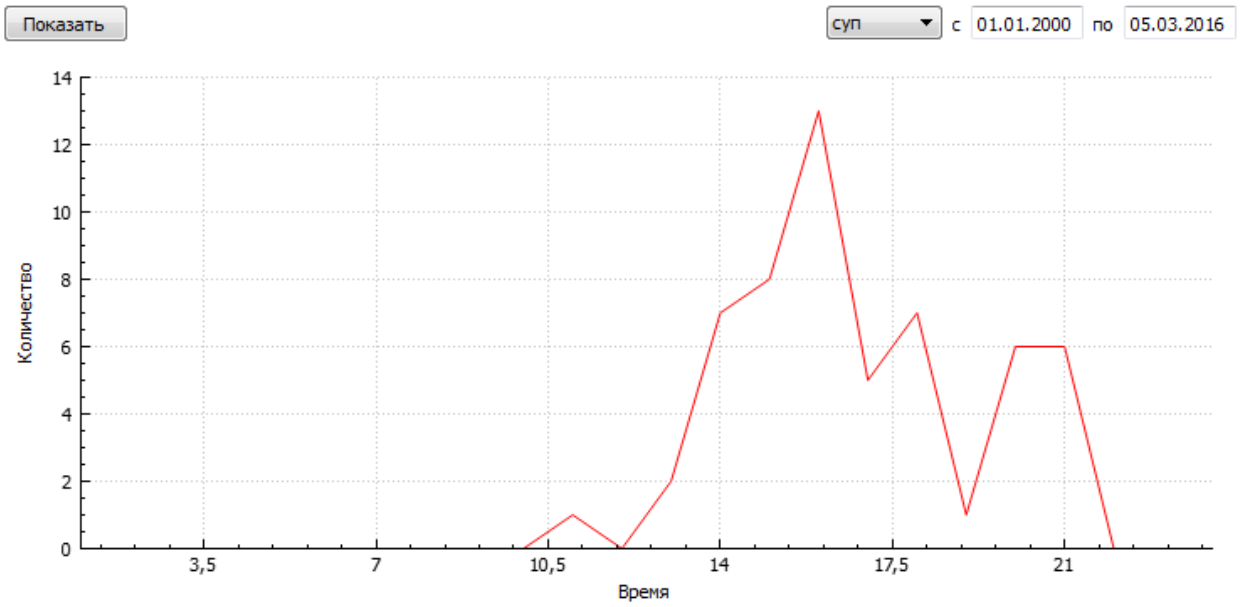


Рисунок 5. График заказов супов

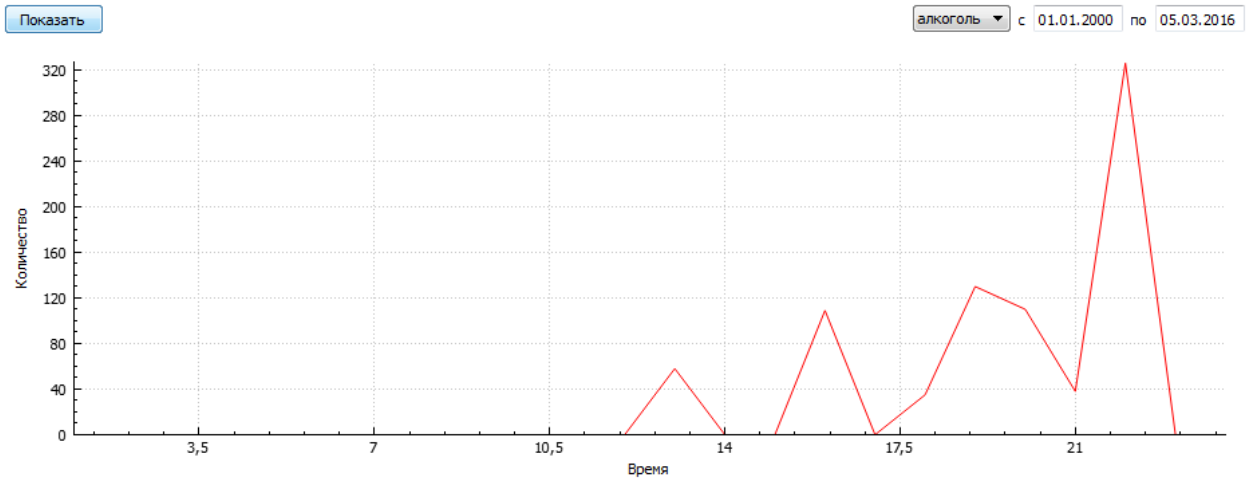


Рисунок 6. График заказов алкоголя

На рисунке 6 приведён график заказов алкогольной продукции. На нём ясно видно, что основной спрос приходится на вечерние часы вплоть до

закрытия заведения. С помощью этого вывода были скорректированы часы работы персонала бара.

Кроме вышесказанного, прогнозы о требуемых в течение следующей недели товарах позволяют получить представление о необходимых закупках в ближайшее время.

Заключение

В настоящей работе было проведено прогнозирование временных рядов, отражающих количественные данные о заказах, сделанных клиентами системы массового обслуживания. Были сделаны выводы о распределении трудовой нагрузки персонала на кухне и в баре. Кроме того, имеется возможность оценить потребность в дополнительных официантах в часы большого количества посетителей и заказов.

В дальнейшем результаты данной работы могут позволить планировать штатное расписание, а также сосредоточиться на минимизации бюджета предприятия, варьируя его основную расходную статью – фонд оплаты труда.

Список литературы

1. Лукашин Ю.П.. *Адаптивные методы краткосрочного прогнозирования временных рядов:* Учеб. пособие. - М.: Финансы и статистика, 2003.-416 с
2. Бокс Дж., Дженкинс Г. *Анализ временных рядов, прогноз и управление*. М.: Мир, 1974. – Вып. 1
3. Akaike, Hirotugu "*A new look at the statistical model identification*". // Transactions on Automatic Control.1974*,* **19**(6): 716–723.
4. Матвеев В. Ф., Ушаков В. Г. *Системы массового обслуживания.* М.: МГУ, 1984. 242стр.
5. Лифшиц А. Л., Мальц Э. А. *Статистическое моделирование систем массового обслуживания.* М., 1974. 248стр.
6. Qt Plotting Widget QCustomPlot www.qcustomplot.com/

Приложение

Файл forecast.m

```
clear all;
close all;
clc;

file = fopen('data\in.csv','r');
X = fscanf(file, '%d');
M = X(1);
p = X(2);
D = X(3);
q = X(4);
X = X(5:end);
N = length(X);

A = arima(p,D,q);
A = estimate(A, X, 'print', false);
[Y YMSE] = forecast(A, M, 'Y0', X);

fclose(file);
file = fopen('data\out.csv','w');
fprintf(file, '%f\n', Y);
fclose(file);
```

Файл dish.h

```
#ifndef DISH_H
#define DISH_H
#include <QString>
#include <QStringList>
#include <QTime>
#include <QDate>
#include <QDebug>
struct Ingr {
    Ingr(QString n, int c) {name = n, count = c;}
    QString name;
    int count;
};
struct Dish {
    Dish(QString t, QString n) {type = t; name = n;}
    Dish(QString t, QString n, QList<Ingr> i) {
        type = t;
        name = n;
        ingredients = i;
    }
    QString type;
    QString name;
    QList<Ingr> ingredients;
};
class DishList
{
private:
    static DishList *instance_;
```

```
public:
    static DishList *getInstance();
    void init();
    QList<Dish> list;
    Dish getDish(QString name);
    QString getType(QString name);
    QStringList getTypes();
};
#endif // DISH_H
```

Файл cuisine.h

```
#ifndef CUISINE_H
#define CUISINE_H
#include <QAbstractTableModel>
#include "dish.h"
const QString dateFormat = QObject::trUtf8("dd-MM-yyyy");
const QString timeFormat = QObject::trUtf8("HH:mm:ss");
struct Order {
    QDate date;
    QTime time;
    QString dish;
    int count;
};
class Cuisine: public QAbstractTableModel
{
private:
    static Cuisine *instance_;
public:
    static Cuisine *getInstance();
    QList<Order> orderList;
    QDate dateFrom, dateTo;
    int rowCount(const QModelIndex &parent = QModelIndex()) const ;
    int columnCount(const QModelIndex &parent = QModelIndex()) const;
    QVariant headerData(int section, Qt::Orientation orientation, int role) const;
    QVariant data(const QModelIndex &index, int role = Qt::DisplayRole) const;
    void clear() {
        orderList.clear();
        dateFrom = QDate(3000, 1, 1);
        dateTo = QDate(2000, 1, 1);
    }
    void addOrder(QString);
};
#endif // CUISINE_H
```

Файл graphform.h

```
#ifndef GRAPHFORM_H
#define GRAPHFORM_H
#include <QWidget>
#include "ui_graphform.h"
#include "qcustomplot/qcustomplot.h"
class GraphForm : public QWidget, public Ui::GraphForm
{
    Q_OBJECT

public:
    explicit GraphForm(QWidget *parent = 0);

    QCustomPlot plot;
};
#endif // GRAPHFORM_H
```

Файл ingrform.h

```
#ifndef INGRFORM_H
#define INGRFORM_H
#include <QWidget>
#include "ui_ingrform.h"
#include "cuisine.h"
class IngrModel : public QAbstractTableModel
{
public:
    int rowCount(const QModelIndex &parent = QModelIndex()) const ;
    int columnCount(const QModelIndex &parent = QModelIndex()) const;
    QVariant headerData(int section, Qt::Orientation orientation, int role) const;
    QVariant data(const QModelIndex &index, int role = Qt::DisplayRole) const;
    QList<Ingr> ingrList;
};
class IngrForm : public QWidget, public Ui::IngrForm
{
    Q_OBJECT

public:
    explicit IngrForm(QWidget *parent = 0);

    IngrModel ingrModel;
    IngrModel forecastModel;
public slots:
    void slotIngrUpdateModel();
    void slotForecast();
};
#endif // INGRFORM_H
```

Файл mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H
#include <QMainWindow>
#include "ui_mainwindow.h"
#include "orderform.h"
#include "graphform.h"
#include "ingrform.h"
#include "cuisine.h"
class MainWindow : public QMainWindow, public Ui::MainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = 0);

private:
    OrderForm orderForm;
    GraphForm graphForm;
    IngrForm ingrForm;
public slots:
    void slotLoadFile();
    void slotImit();
    void slotPlot();
};
#endif // MAINWINDOW_H
```

Файл orderform.h

```
#ifndef ORDERFORM_H
#define ORDERFORM_H
#include <QWidget>
#include "ui_orderform.h"
class OrderForm : public QWidget, public Ui::OrderForm
{
    Q_OBJECT

public:
    explicit OrderForm(QWidget *parent = 0);
};
#endif // ORDERFORM_H
```