МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Кафедра вычислительных технологий**

**КУРСОВАЯ РАБОТА**

**ПРИМЕНЕНИЕ МЕТОДОВ ИНТЕЛЛЕКТУАЛЬНОГО АНАЛИЗА ДАННЫХ ДЛЯ ЗАДАЧ МЕДИЦИНСКОЙ ДИАГНОСТИКИ**

Работу выполнил М.И. Ващанов

(подпись, дата) (инициалы, фамилия)

Факультет компьютерных технологий и прикладной математики 3 курс

Направление 02.03.02 – «Фундаментальная информатика и информационные технологии»

Научный руководитель

доц., канд.т.н. Т.А. Приходько

(подпись, дата) (инициалы, фамилия)

Нормоконтролер

ст.преп., канд.т.н., Е.Е. Полупанова

(подпись, дата) (инициалы, фамилия)

Краснодар 2017

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc501233363)

[1 Основные понятия задачи классификации 5](#_Toc501233364)

[1.1. Формальная постановка задачи классификации 5](#_Toc501233365)

[1.2. Линейная классификация 5](#_Toc501233366)

[1.3. Метрическая классификация. Классификация методом k ближайших соседей 6](#_Toc501233367)

[1.4. Логическая классификация 7](#_Toc501233368)

[1.5.Наивный Байесовский классификатор 8](#_Toc501233369)

[2 Имплементация методов классификации 11](#_Toc501233370)

[2.1. Инструменты анализа данных 11](#_Toc501233371)

[2.2. Метод k ближайших соседей 13](#_Toc501233372)

[2.3. Наивный Байесовский классификатор 14](#_Toc501233373)

[3 Исследование медицинских данных 15](#_Toc501233374)

[3.1. Описание исходных данных 15](#_Toc501233375)

[3.2. Алгоритм решения задачи. 16](#_Toc501233376)

[3.3. Первичный визуальный анализ. 17](#_Toc501233377)

[3.4. Обучение классификаторов 19](#_Toc501233378)

[4 Анализ полученных результатов 20](#_Toc501233379)

[Заключение 23](#_Toc501233380)

[Список литературы 24](#_Toc501233381)

# ВВЕДЕНИЕ

Классификация — один из разделов машинного обучения, посвященный решению следующей задачи. Имеется множество объектов (ситуаций), разделённых некоторым образом на классы. Задано конечное множество объектов, для которых известно, к каким классам они относятся. Это множество называется обучающей выборкой. Классовая принадлежность остальных объектов не известна. Требуется построить алгоритм, способный классифицировать произвольный объект из исходного множества. В машинном обучении задача классификации относится к разделу обучения с учителем. Классификацию сигналов и изображений называют также распознаванием образов. Данные об объектах классификации могут представляться в разных формах. Наиболее частым случаем является представление объекта через признаковое описание, однако существуют и другие виды представления данных (матрица расстояний, временной ряд, изображение или видеоряд).

Примерами практических задач классификации являются задачи медицинской диагностики, где на вход подаётся информация о пациенте (симптомы заболевания, тяжесть состояния, результаты обследований), а на выходе может быть вид заболевания, рекомендуемое лечение, риск осложнений и пр. Также методами интеллектуального анализа данных могут быть обнаружены синдромы (совокупности симптомов) заболеваний.

Ценность такого рода систем в том, что они способны анализировать и обобщать огромное количество прецедентов — возможность, недоступная специалисту-врачу. Также благодаря высокой скорости получения ответа от уже обученной модели врачам легче быстро принимать решения в неотложных случаях.

Целью работы является изучение методов интеллектуального анализа данных в части методов классификации и нахождение наилучшего алгоритма для задачи постановки диагноза диабета пациенту.

Курсовая работа состоит из пяти глав.

Первая глава работы содержит общие теоретические сведения о задаче классификации и алгоритмах её решения.

Вторая глава данной работы посвящена реализации алгоритмов классификации.

В третьей главе исследуются медицинские данные.

Четвертая глава содержит информацию о настройке классификаторов под конкретную задачу.

В пятой главе проводится анализ полученных результатов.

## Основные понятия задачи классификации

### Формальная постановка задачи классификации

Пусть X – множество описаний объектов, Y – конечное множество меток классов. Существует неизвестная целевая зависимость — отображение , значения которой известны только на объектах конечной обучающей выборки Требуется построить алгоритм , способный классифицировать произвольный объект [5]

Существуют и различные виды классов:

1. Двухклассовая классификация. Наиболее простой в техническом отношении случай, который служит основой для решения более сложных задач.
2. Многоклассовая классификация. Когда число классов достигает многих тысяч (например, при распознавании иероглифов или слитной речи), задача классификации становится существенно более трудной.
3. Непересекающиеся классы.
4. Пересекающиеся классы. Объект может относиться одновременно к нескольким классам.
5. Нечёткие классы. Требуется определять степень принадлежности объекта каждому из классов, обычно это действительное число от 0 до 1.

### Линейная классификация

Линейный классификатор – алгоритм классификации, основанный на построении линейной разделяющей поверхности. В случае двух классов разделяющей поверхностью является гиперплоскость, которая делит пространство признаков на два полупространства. В случае большего числа классов разделяющая поверхность кусочно-линейна.

Пусть объекты описываются *n* числовыми признаками . Тогда пространство признаковых описаний объектов есть . Пусть  конечное множество меток классов. Рассмотрим задачу бинарной классификации. Пусть .

Линейным классификатором называется алгоритм классификации    вида , где  – вес i-го признака,  – порог принятия решения,  – вектор весов,  – скалярное произведение признакового описания объекта на вектор весов. Предполагается, что искусственно введён «константный» нулевой признак: .[5]

### 1.3. Метрическая классификация. Классификация методом k ближайших соседей

Метрические классификаторы опираются на гипотезу компактности, которая предполагает, что схожие объекты чаще лежат в одном классе, чем в разных. Это означает, что граница между классами имеет достаточно простую форму, и классы образуют компактно локализованные области в пространстве объектов. В метрических алгоритмах классифицируемый объект может описываться не набором признаков, а непосредственно вектором расстояний до остальных объектов обучающей выборки. В таких случаях говорят также о беспризнаковом распознавании.

Например, сходство текстов, химических формул и аминокислотных последовательностей гораздо проще измерять непосредственно, чем переходя к признаковым описаниям. В экспертных системах важно не только классифицировать объекты, но и выдавать пользователю объяснение предлагаемой классификации. В методе ближайшего соседа такие объяснения выглядят весьма разумно: «Объект x отнесён к классу C потому, что к этому же классу относился близкий объект обучающей выборки». Такая «прецедентная» логика хорошо понятна экспертам во многих предметных областях (медицине, геологии, юриспруденции).

Метод k ближайших соседей — простейший [метрический классификатор](http://www.machinelearning.ru/wiki/index.php?title=%D0%9C%D0%B5%D1%82%D1%80%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%BA%D0%BB%D0%B0%D1%81%D1%81%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%82%D0%BE%D1%80), основанный на оценивании [сходства](http://www.machinelearning.ru/wiki/index.php?title=%D0%A1%D1%85%D0%BE%D0%B4%D1%81%D1%82%D0%B2%D0%BE&action=edit) объектов. Классифицируемый объект относится к тому классу, которому принадлежат k ближайших к нему объектов обучающей выборки. [2, 5]

### 1.4. Логическая классификация

Решающие деревья воспроизводят логические схемы, позволяющие получить окончательное решение о классификации объекта с помощью ответов на иерархически организованную систему вопросов. Причём вопрос, задаваемый на последующем иерархическом уровне, зависит от ответа, полученного на предыдущем уровне. Подобные логические модели издавна используются в ботанике, зоологии, минералогии, медицине и других областях.

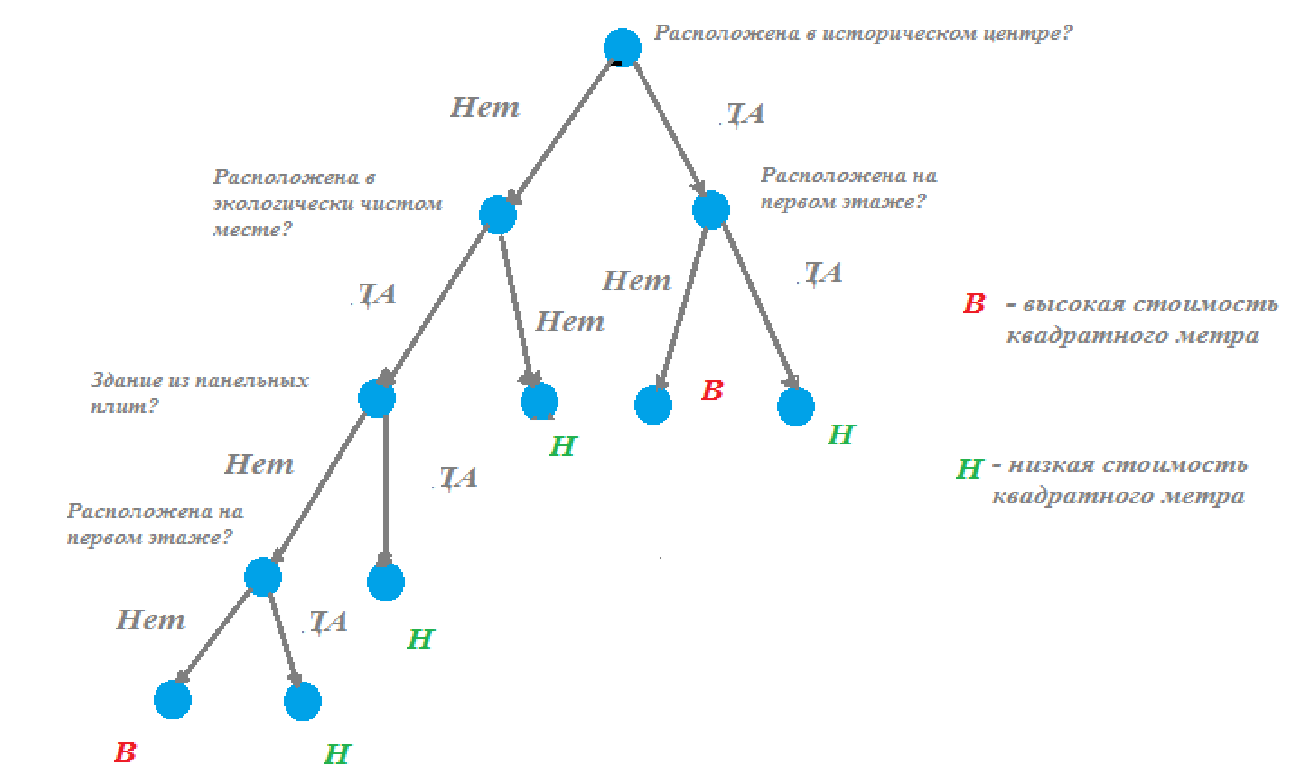


Рис. 1 Пример решающего дерева.

На рис.1 изображён пример решающего дерева для задачи грубой оценки стоимости 1 квадратного метра жилья в некотором городе.

Обучение решающего дерева происходит по следующему алгоритму:

В корне дерева — рассматриваем всю [обучающую выборку](http://ru.learnmachinelearning.wikia.com/wiki/%D0%90%D0%BD%D0%B0%D0%BB%D0%B8%D0%B7_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85?redlink=1&action=edit&flow=create-page-article-redlink).

1. Проверить критерий останова алгоритма. Если он выполняется, выбрать для узла выдаваемый прогноз, что можно сделать несколькими способами.
2. Иначе требуется разбить множество на несколько не пересекающихся. В общем случае в вершине t задаётся решающее правило, принимающее некоторый диапазон значений. Этот диапазон разбивается на непересекающихся множеств объектов, , где — количество потомков у вершины, а каждое — это множество объектов, попавших в i-го потомка
3. Множество в узле разбивается согласно выбранному правилу, для каждого узла алгоритм запускается рекурсивно.

Основным недостатком алгоритма является тенденция к переобучению. Для борьбы с переобучением часто вводят ограничения на глубину дерева и/или на минимальное количество объектов в листе дерева.

### 1.5.Наивный Байесовский классификатор

Байесовский подход к классификации основан на теореме, утверждающей, что если плотности распределения каждого из классов известны, то искомый алгоритм можно выписать в явном аналитическом виде. Более того, этот алгоритм оптимален, то есть обладает минимальной вероятностью ошибок.

Байесовские классификаторы основываются на формуле Байеса:

, (1)

где P(y|x) – апостериорная вероятность данного класса c (т.е. данного значения целевой переменной) при данном значении признака x, P(y) – априорная вероятность данного класса, P(x|y) – правдоподобие, т.е. вероятность данного значения признака при данном классе, P(x) – априорная вероятность данного значения признака. Так как решается задача классификации, а не непосредственное вычисление вероятности, то можно пренебречь вычислением значения P(x). Это объясняется тем, что есть необходимость только в нахождении такого , что . Из формулы (1) видно, что значение P(x) не влияет на значение y, а это значит, что вычислять его нет необходимости.

На практике плотности распределения классов, как правило, не известны. Их приходится оценивать (восстанавливать) по обучающей выборке. В результате байесовский алгоритм перестаёт быть оптимальным, так как восстановить плотность по выборке можно только с некоторой погрешностью. Чем короче выборка, тем выше шансы подогнать распределение под конкретные данные и столкнуться с эффектом переобучения.[3]

Байесовский подход к классификации является одним из старейших, но до сих пор сохраняет прочные позиции в теории распознавания. Он лежит в основе многих достаточно удачных алгоритмов. Примером такого алгоритма может послужить EM-алгоритм, который служит для нахождения оценок максимального правдоподобия и для разделения смесей распределений.

Наивный байесовский классификатор — специальный частный случай байесовского классификатора, основанный на дополнительном предположении, что объекты описываются n статистически независимыми признаками. Основные преимущества наивного байесовского классификатора — простота реализации и низкие вычислительные затраты при обучении и классификации. В тех редких случаях, когда признаки действительно независимы (или почти независимы), наивный байесовский классификатор (почти) оптимален.[5]

Достоинства байесовского классификатора:

* хорошо работает на практике, когда данные имеют вероятностную природу;
* простая реализация;
* скорость работы;
* разделяет объекты достаточно простыми, но нетривиальными разделяющими поверхностями (в случае нормальных распределений) [4].

Алгоритм построения наивного байесовского классификатора.

Данный алгоритм основывается на предположении, что все признаки независимы и распределены в соответствии с нормальным законом распределения.

Этап обучения модели состоит в вычислении матожидания и среднеквадратичного отклонения каждого признака в каждом классе.

Этап получения предсказания для объекта x состоит из следующих шагов:

* 1. Для всех классов выполняются шаги 2-4
  2. Вычисляется априорная вероятность появления класса по классическому определению вероятности.
  3. Вычисляется суммарной апостериорная вероятность признаков x при условии класса y, т.е. вероятность получить x при нормальном распределении с матожиданием и среднеквадратичным отклонением, вычисленными на этапе обучения модели.
  4. Возвращается класс с максимальным значением апостериорной вероятности.

## Имплементация методов классификации

### 2.1. Инструменты анализа данных

Одним из наиболее популярных языков программирования для анализа данных является Python. Для него создано большое количество библиотек и фреймворков. В работе используется интерактивная оболочка IPython Jupiter Notebook. Данная оболочка поддерживает встроенный режим работы с самой популярной библиотекой визуализации для Python matplotlib. Также в ней присутствует функция разбиения исходного кода на независимые блоки, вследствие чего появляется возможность запуска программы “по частям”. Это является очень полезной функцией, потому что можно, например, один раз обучить некоторую модель в одном блоке кода, а анализ и тестирование её вывести в другие блоки, которые можно изменять, перезапускать, вручную менять последовательность их исполнения. Этот функционал получилось реализовать, потому что Python является интерпретируемым языком (т.е. исполняется построчно).

Для визуализации в работе используются библиотеки matplotlib и seaborn.

Хранение данных осуществляется в оперативной памяти с помощью библиотеки pandas. Pandas — программная библиотека на языке Python для обработки и анализа данных. Работа pandas с данными строится поверх библиотеки NumPy, являющейся инструментом более низкого уровня. Pandas предоставляет специальные структуры данных и операции для манипулирования числовыми таблицами и временными рядами. Название библиотеки происходит от эконометрического термина «панельные данные», используемого для описания многомерных структурированных наборов информации. Основные возможности библиотеки:

* объект DataFrame для манипулирования индексированными массивами двумерных данных;
* инструменты для обмена данными между структурами в памяти и файлами различных форматов;
* встроенные средства совмещения данных и способы обработки отсутствующей информации;
* переформатирование наборов данных, в том числе создание сводных таблиц;
* срез данных по значениям индекса, расширенные возможности индексирования, выборка из больших наборов данных;
* вставка и удаление столбцов данных;
* возможности группировки позволяют выполнять трёхэтапные операции типа «разделение, изменение, объединение»;
* слияние и объединение наборов данных;
* иерархическое индексирование позволяет работать с данными высокой размерности в структурах меньшей размерности;
* работа с временными рядами: формирование временных периодов и изменение интервалов и т. д.

Для реализации алгоритмов используется библиотека NumPy. NumPy — это библиотека с открытым исходным кодом для языка программирования Python. Основной возможностью этой библиотеки является поддержка многомерных массивов (включая матрицы) и поддержка высокоуровневых математических функций, предназначенных для работы с многомерными массивами. Эта библиотека использует параллельные вычисления, и содержит алгоритмы для решения задач линейной алгебры. Numpy входит в библиотеку для научных вычислений SciPy.

Возможности пакета SciPy достаточно обширны. С помощью этой библиотеки можно:

* искать минимумы и максимумы функций;
* вычислять интегралы функций;
* обрабатывать сигналы и изображения;
* работать с генетическими алгоритмами;
* решать обыкновенные дифференциальные уравнения и др.

### 2.2. Метод k ближайших соседей

Метод k ближайших соседей реализован в файле knn.py в виде класса KNNClassifier с конструктором \_\_init\_\_(k, distance\_func), методами fit(X, y), predict(X), aggregate(neighbors\_targets), \_predict\_x(x).

В конструктор передается параметр k – количество ближайших соседей (по умолчанию k=5), и параметр distance\_func – используемая функция расстояния для вычисления расстояния между двумя объектами (по умолчанию евклидова метрика).

В метод fit передаются данные X и их метки y. Они сохраняются в полях класса.

Метод predict принимает список объектов X, которые нужно классифицировать, а возвращает список их меток. Для каждого объекта x из X вызывается метод \_predict\_x(x), который возвращает метку класса для этого объекта.

Метод \_predict\_x(x) работает следующим образом:

1. Формируется список distance, i-ый элемент которого равен расстоянию между i-ым объектом в X и объектом x.
2. Формируется список neighbours, элементами которого являются пары {расстояние, метка} для каждого объекта из X и каждой метки из y.
3. Формируется список neighbors\_targets, элементами которого являются метки k ближайших соседей объекта x.
4. Вызывается функция aggregate(neighbors\_targets), которая возвращает самую часто встречающуюся метку. Если таких меток несколько, то возвращается любая из них.

### 2.3. Наивный Байесовский классификатор

Наивный Байесовский классификатор реализован в файле bayes.py в виде класса NaiveBayes с методами fit(X, y), \_calculate\_likelihood(mean, var, x), \_calculate\_prior(c), \_classify(sample), predict(X).

Метод fit(X, y) обучает классификатор. Производит вычисление матожидания и среднеквадратичного отклонения каждого признака в каждом классе.

Метод \_calculate\_likelihood(mean, var, x) оценивает вероятность получить x при нормальном распределении с матожиданием *mean* среднеквадратичным отклонением *var.*

Метод \_calculate\_prior(с) подсчитывает априорную вероятность класса *c*.

Метод \_classify(sample) осуществляет классификацию объекта *sample* в соответствии с п. 1.5алгоритма получения предсказания для байесовского классификатора.

Метод predict(X) получает на вход список объектов X, метки классов которых необходимо предсказать. Возвращает список меток, полученный путём вызова метода \_classify от каждого объекта из списка X.

## Исследование медицинских данных

### 3.1. Описание исходных данных

В работе исследуется набор данных о больных диабетом. Данные представляются в виде таблицы формата csv, в каждой строке которой записана информация об одном человеке. Целью анализа в данной задаче является построение модели, способной предсказывать наличие сахарного диабета у пациента по признакам из данного набора данных.

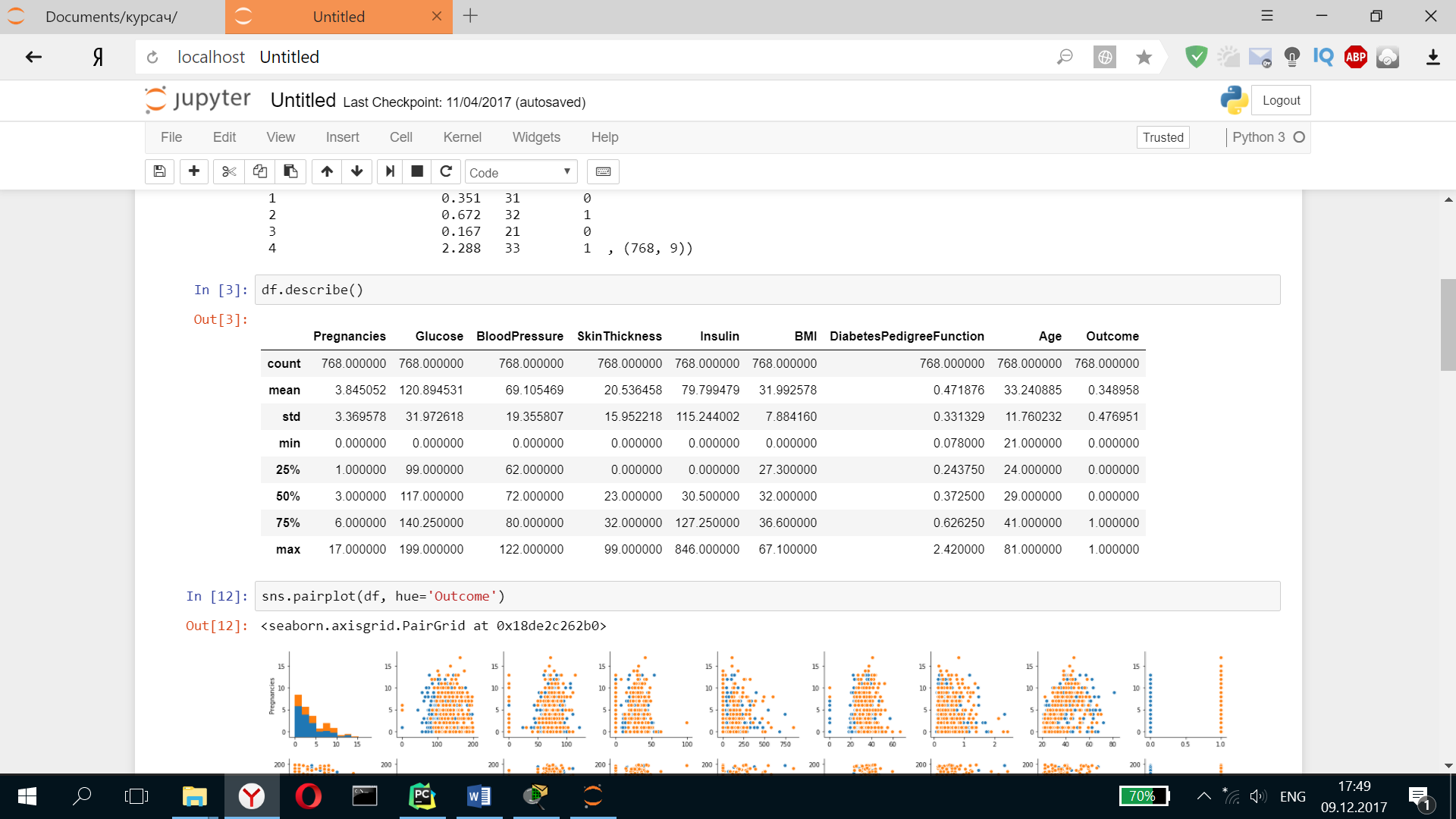


Рисунок 1 – краткая статистика данных

На рис. 1 показана таблица с краткими статистическими характеристиками исходных данных.

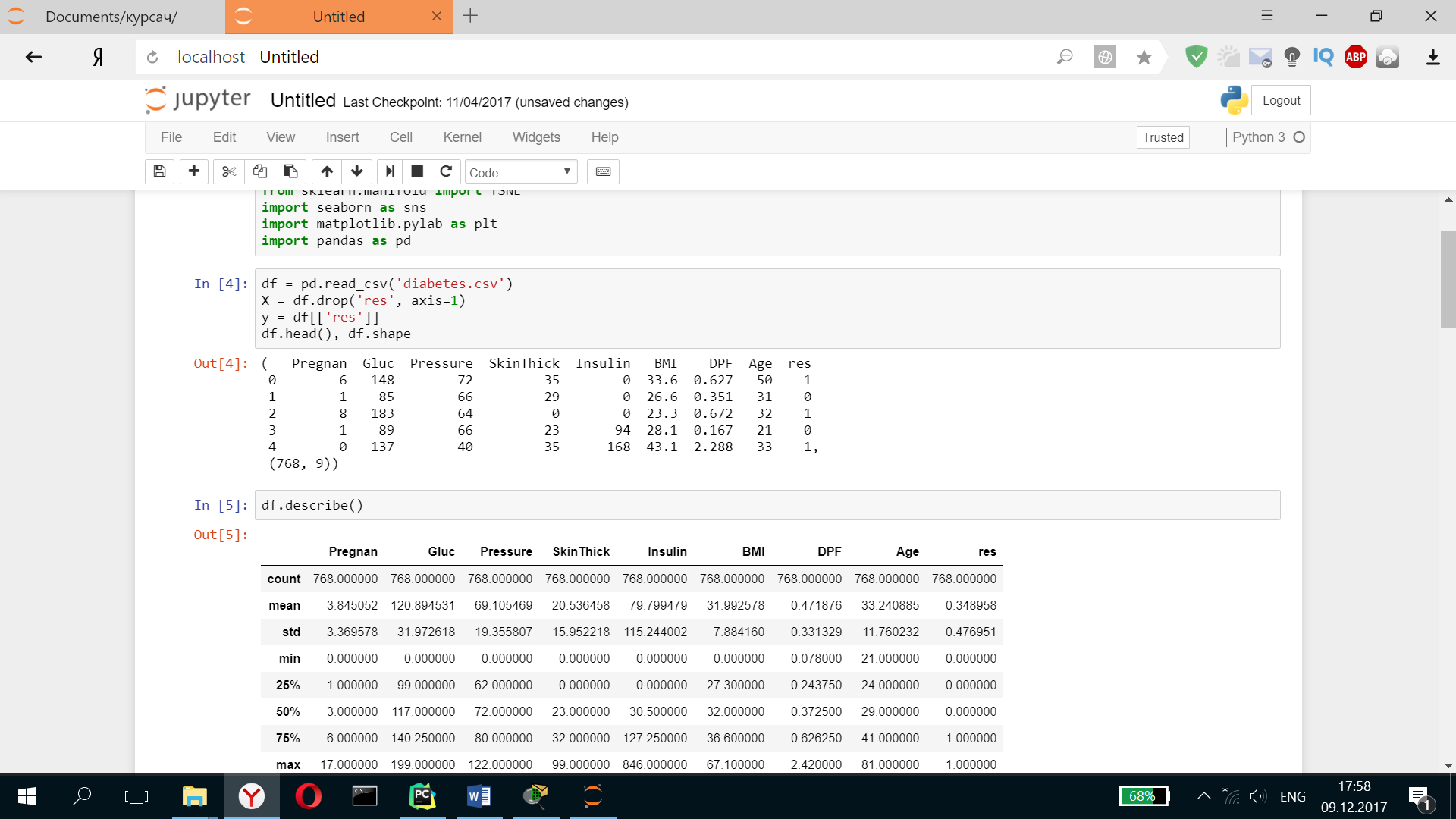


Рисунок 2 – Фрагмент исходной таблицы данных

На рис. 2 показаны первые 5 строк таблицы. Из таблицы видно, что один человек описывается девятью признаками, один из которых (признак res) является меткой класса.

### Алгоритм решения задачи

Алгоритм решения поставленной задачи состоит из следующих шагов:

1. С помощью визуального анализа выделить скоррелированные признаки, оценить взаимосвязи всех признаков с целевым. Использовать для этого графики scatter plot и boxplot.
2. Настроить классификаторы под поставленную задачу.
3. Оценить точность классификаторов по кросс-валидации и выбрать наилучший.
4. Указать, возможна ли доработка и улучшение классификатора, полученного на шаге 3.

### Первичный визуальный анализ

Прежде чем работать с данными, необходимо как можно глубже понять их природу. Для этого часто используется визуальный анализ. Визуализация производится с помощью библиотеки seaborn. Реализация процесса визуализации находится в файле visual.ipynb.

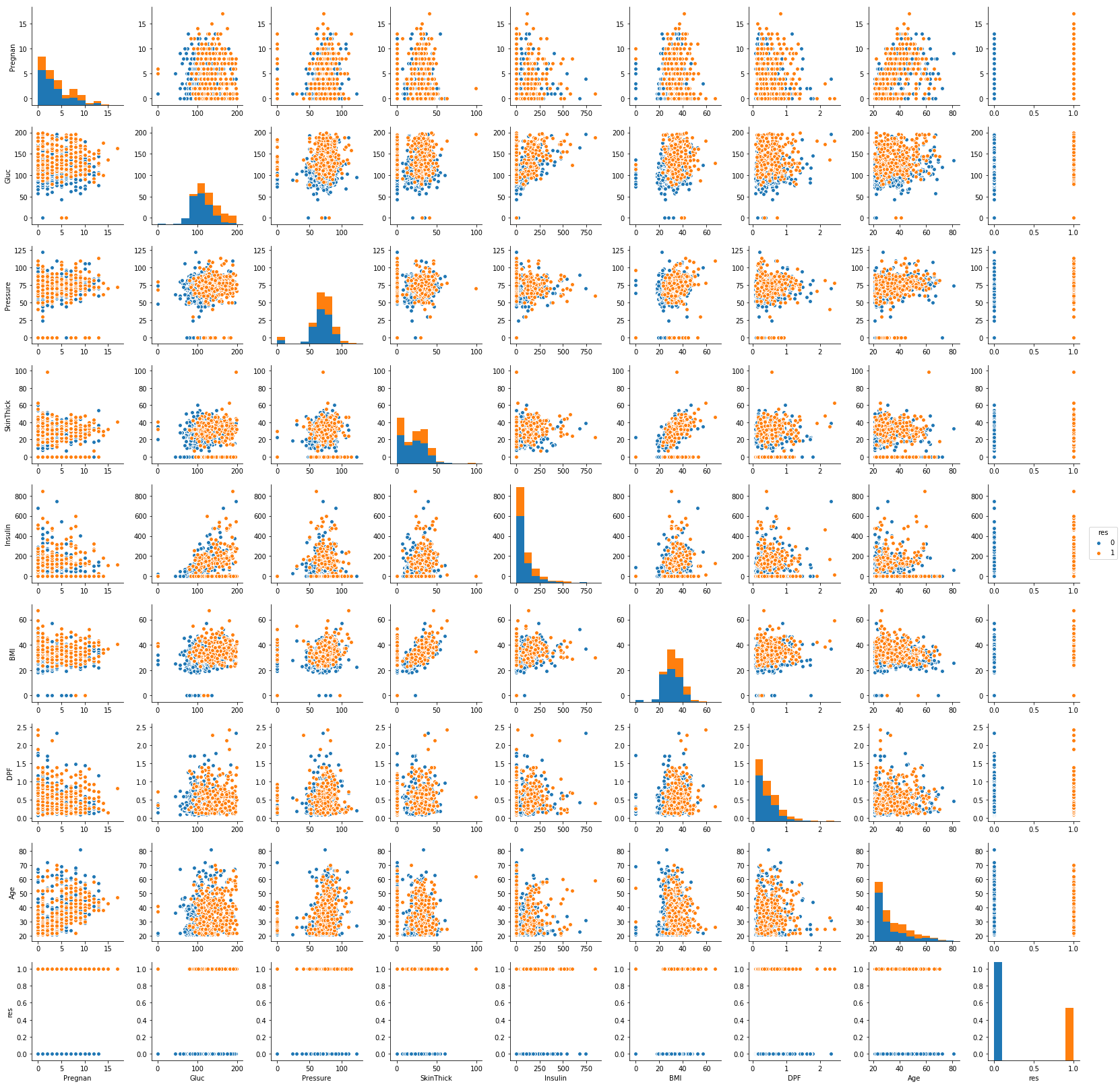


Рисунок 3 – графики scatter plot

На рис. 3 приведены графики scatter plot для всех пар признаков. Оранжевыми точками обозначены больные диабетом, синими – здоровые люди. По диагонали расположены распределения признаков. По этому графику можно визуально определить наличие корреляции между парой признаков. Визуально небольшая корреляция видна только между признаками BMI и SkinThick. Это значит, что признаки достаточно сильно независимы друг от друга, отсутствуют признаки, которые можно вычислить через другие признаки.

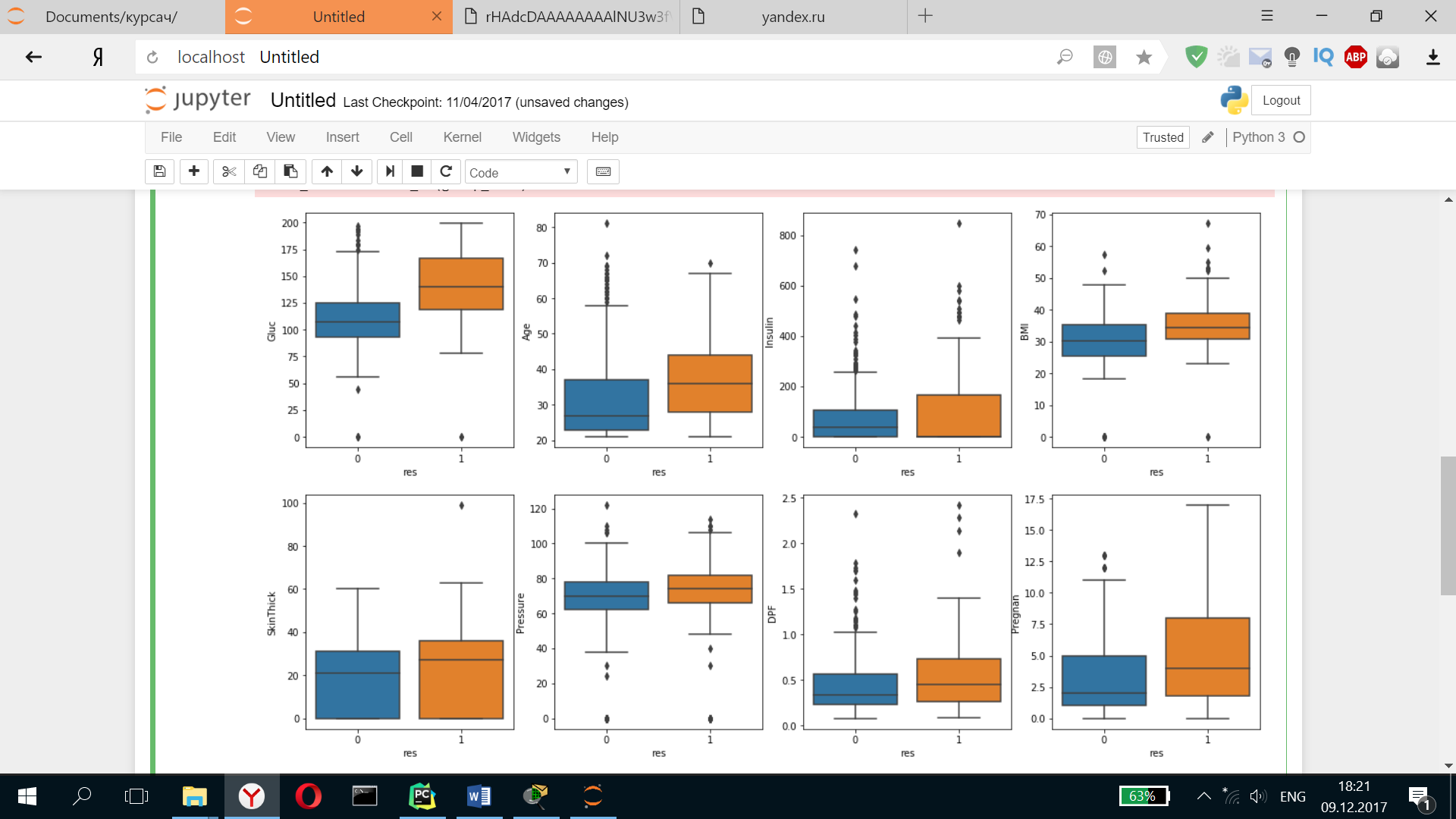


Рисунок 4 – графики boxplot

На рис. 4 показаны графики boxplot. По этим графикам можно судить о сходстве и различии некоторых параметров распределений признаков между здоровыми и больными людьми. Например, визуально видно, что средний уровень глюкозы в крови намного выше у больных диабетом, чем у здоровых.

### Обучение классификаторов

Обучение модели является нетривиальной задачей, так как модель может обладать параметрами, которые необходимо подбирать под конкретную задачу. Такие параметры называются гиперпараметрами. Примером гиперпараметра может служить значение k в алгоритме k ближайших соседей. Для нахождения гиперпараметров существуют следующие стратегии:

* Подбор гиперпараметра вручную
* Подбор гиперпараметра по кросс-валидации

Кросс-валидация – метод оценки модели и её поведения на независимых данных. При оценке модели, имеющиеся в наличии данные, разбиваются на k частей. Затем на k-1 частях данных производится обучение модели, а оставшаяся часть данных используется для тестирования. Эта процедура повторяется k раз. В итоге каждая из k частей используется для тестирования. В результате получается оценка эффективности выбранной модели с наиболее равномерным использованием имеющихся данных [3].

Применяя стратегию кросс-валидации KFold с 7 частями к методу k ближайших соседей, получаем, что оптимальным значением гиперпараметра k для этого набора данных является 17. Реализация данной стратегии находится в файле analyze.ipynb.

У наивного байесовского классификатора отсутствуют гиперпараметры. Следовательно, он не нуждается в специальной настройке под конкретную задачу.

## Анализ полученных результатов

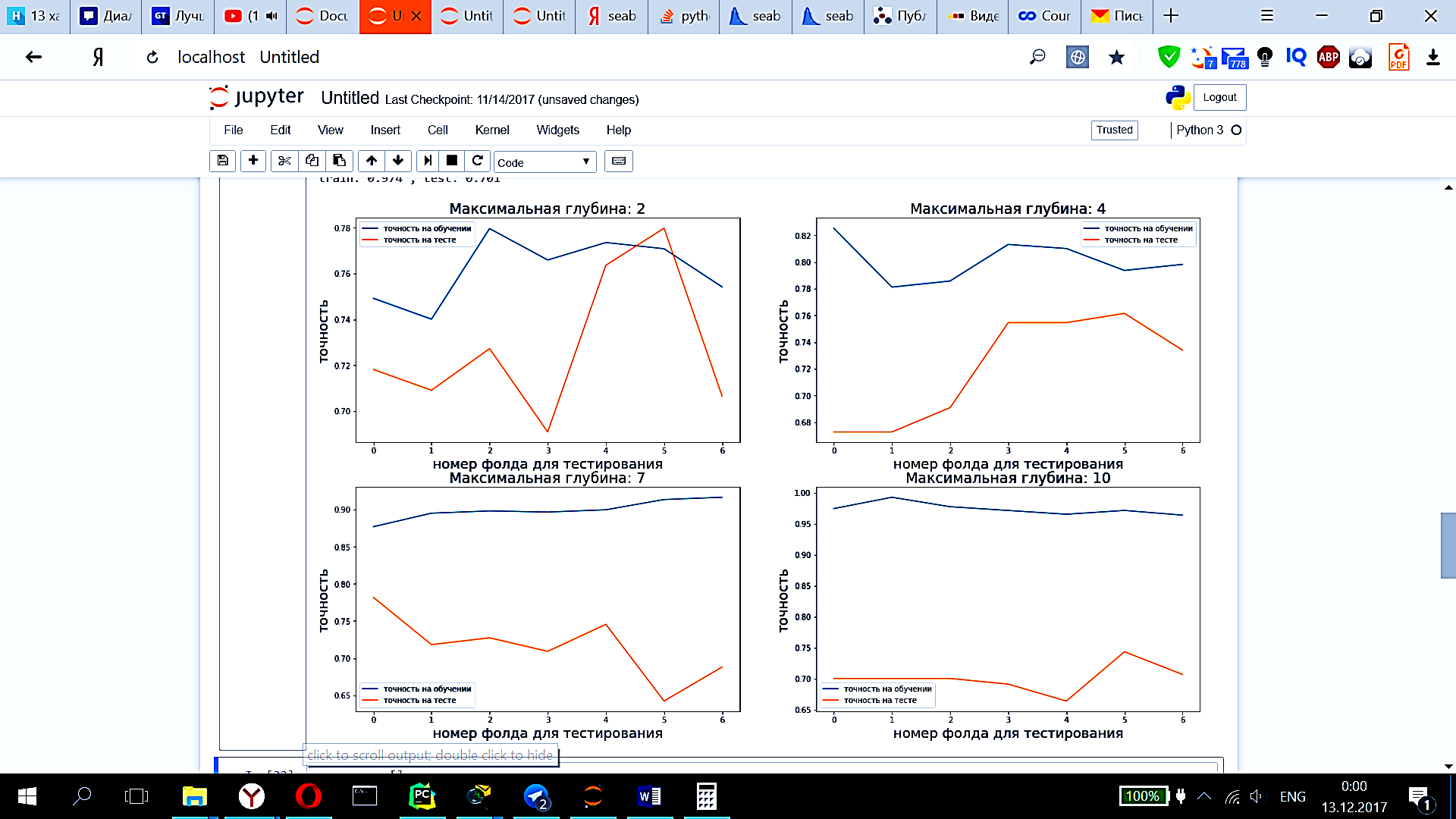
В результате применения алгоритмов классификации к выборке медицинских данных была получена следующая информация о средней точности классификаторов:

Таблица1. Средняя точность классификаторов.

|  |  |
| --- | --- |
| Название классификатора | Средняя точность |
| k ближайших соседей | 78,4% |
| Наивный байесовский классификатор | 75,8% |
| Решающее дерево | 74,6% |
| SVM с линейным ядром | 61% |

Наиболее успешным алгоритмом оказался метод k ближайших соседей со средней точностью 78,4%. Этот высокий показатель можно объяснить сходством логики работы алгоритма и логики принятия решения врачом. Практическим плюсом этого алгоритма является возможность его доработки таким образом, чтобы он выводил список ближайших соседей. Это поможет специалисту вспомнить похожие случаи из его практики и сильно сузить область возможных решений.

Решающее дерево тоже показало неплохой результат. Его практическое преимущество заключается в том, что его можно доработать так, чтобы оно могло выводить вопросы в узлах дерева [2]. А врач, проанализировав эти вопросы мог улучшить свои методики постановки диагнозов. Особое внимание при этом надо обращать на те вопросы, которые идут первыми, так как именно они чаще всего позволяют наиболее сильно сократить дальнейший поиск. Это объясняется тем, что критериями выбора того или иного условия в узле дерева чаще всего становятся индекс Джини или кросс-энтропийный критерий [2] [3].



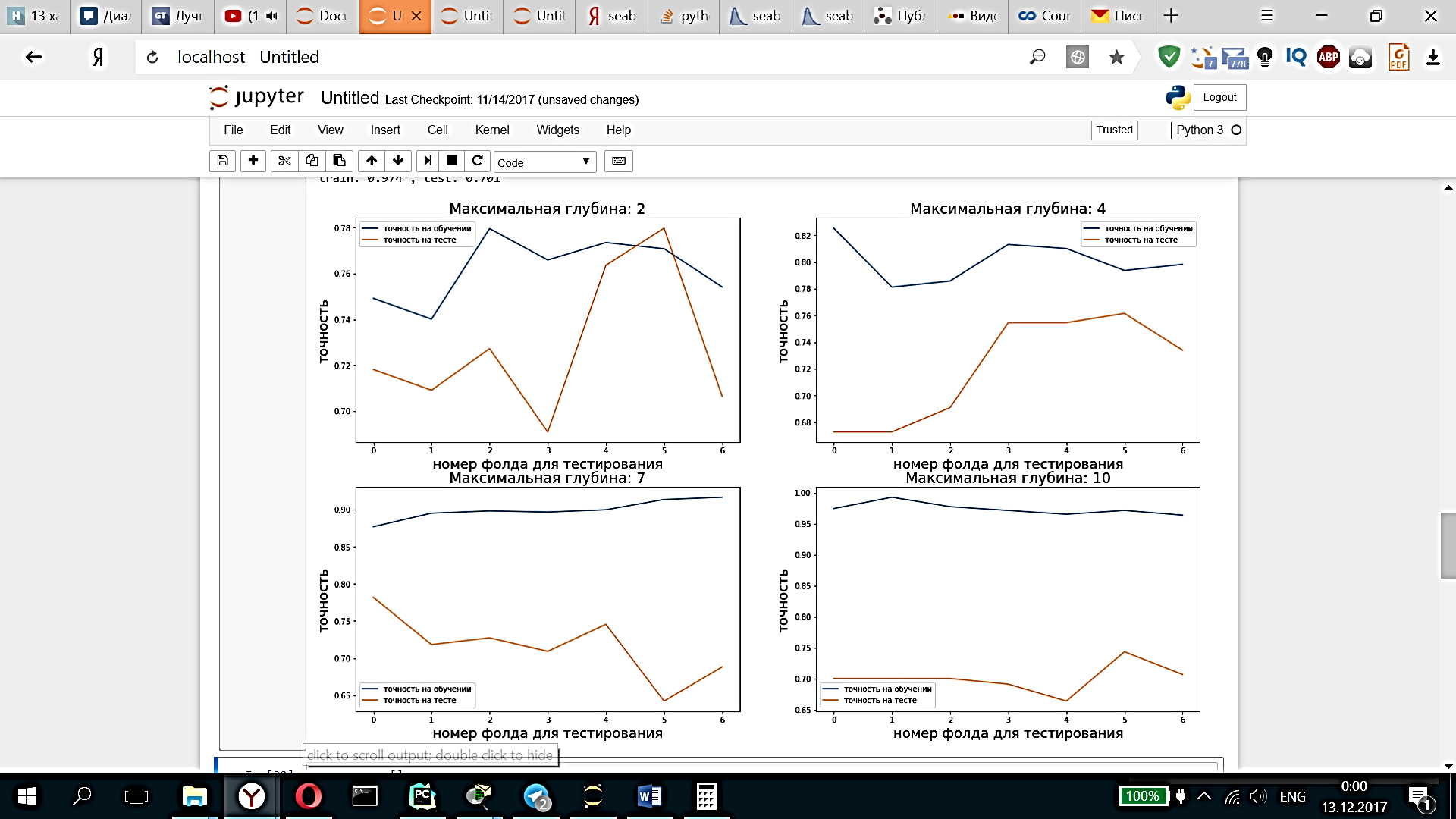


Рисунок 5 – сравнение точности на обучающей и тестовой выборках для различных значений максимальной глубины дерева в алгоритме решающего дерева.

На рисунке 5 видно, что уже при максимальной глубине 7 дерево начинает переобучаться и подгоняться под обучающую выборку, а точность на тестовой выборке уменьшается.

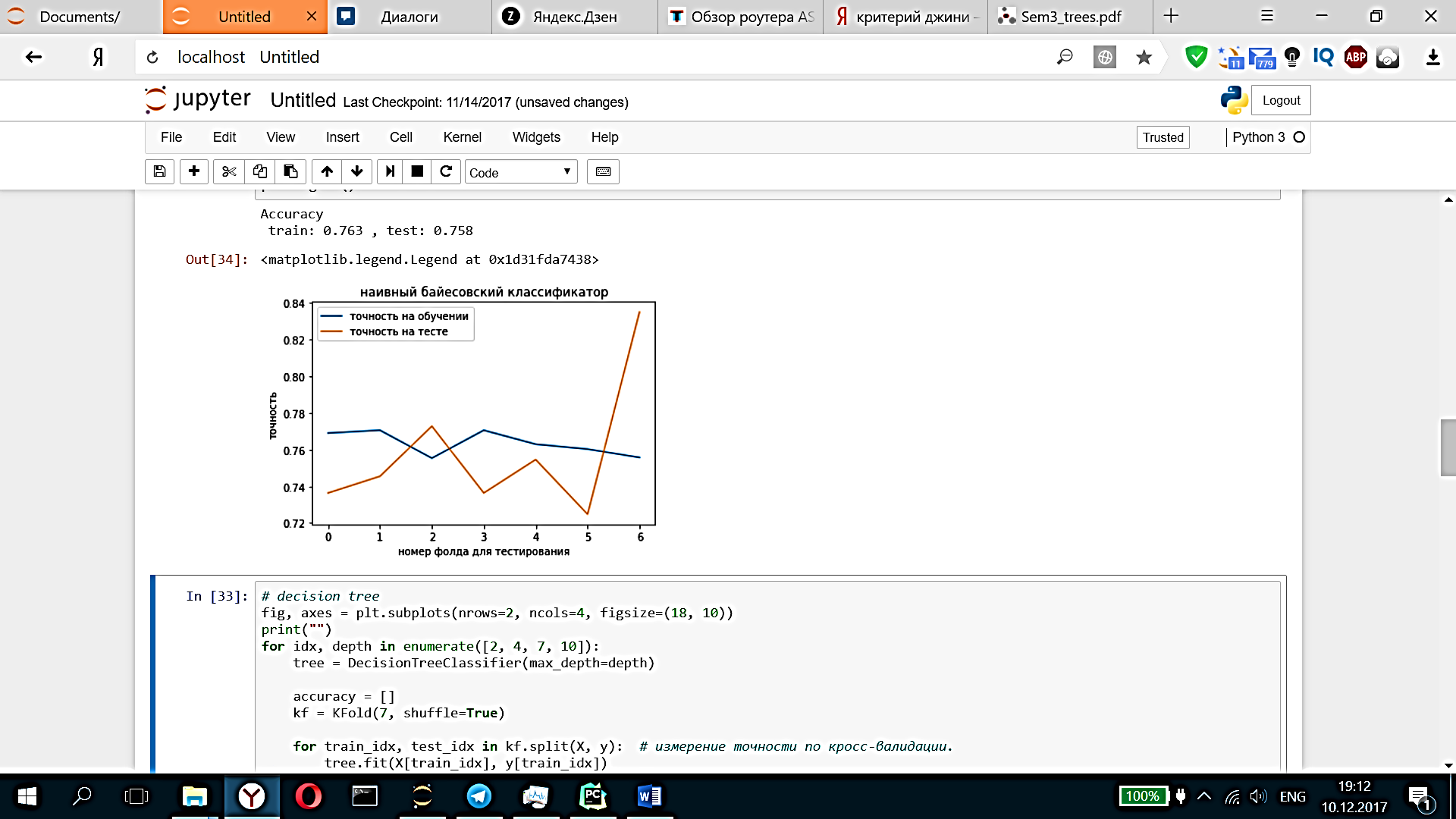


Рисунок 6 – сравнение точности на обучающей и тестовой выборках наивного байесовского классификатора.

На рисунке 6 видно, что байесовский классификатор работает хорошо: он не переобучился и точность его предсказания достаточно высока. Следует вспомнить, что в результате первичного визуального анализа была обнаружена пара признаков BMI и SkinThick, которые не являются независимыми. Также вполне возможно, что существуют зависимости не только между двумя признаками, а между тремя и более признаками. Однако, это не помешало данному алгоритму показать достойные результаты.

Метод линейной классификации SVM показал самый низкий результат. Это означает, что данные оптимально не разделяются (n-1)-мерной гиперплоскостью. Однако алгоритм SVM потенциально может дать хороший результат, если нелинейно преобразовать исходные данные или использовать нелинейные ядра [1] [2] [5].

# Заключение

В ходе работы были изучены следующие алгоритмы классификации: метод k ближайших соседей, решающее дерево, наивный байесовский классификатор, линейный классификатор. В задаче постановки диагноза диабета наиболее эффективным оказался метод k ближайших соседей. Этот алгоритм достаточно хорошо исследован и может быть улучшен. Особое внимание при дальнейшем исследовании необходимо уделить функции расстояния, специализировав её под конкретную задачу.

Также можно поэкспериментировать с методами feature engeneering для нахождения новых информативных признаков в данных из уже существующих.

# Список литературы

1 Hastie, T., Tibshirani R., Friedman J. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. — 2nd ed. — Springer-Verlag, 2009. — 746 p. — ISBN 978-0-387-84857-0

2 Видеолекции курса «Машинное обучение». 2016 [электронный ресурс]. – URL: https://yandexdataschool.ru/edu-process/courses/machine-learning (дата обращения 15.11.2017).

3 Онлайн-курс «Обучение на размеченных данных». 2017 [электронный ресурс]. – URL: <https://www.coursera.org/learn/supervised-learning/home/welcome> (дата обращения 10.10.2017).

4 Дьяконов, A. Г. Анализ данных, обучение по прецедентам, логические игры, системы WEKA, RapidMiner и MatLab (практикум на ЭВМ кафедры математических методов прогнозирования). — МАКСПресс, 2010. — 278 с.

5 Статьи с сайта http://www.machinelearning.ru. [электронный ресурс] (дата обращения 14.12.2017).

# ПРИЛОЖЕНИЕ

Листинг программы:

bayes.py:

import numpy as np

class NaiveBayes():

"""Наивный Байесовский классификатор."""

def fit(self, X, y):

self.X, self.y = X, y

self.classes = np.unique(y)

self.parameters = []

# Вычисление матожидания и среднеквадратичного отклонения каждого признака в каждом классе.

for i, c in enumerate(self.classes):

X\_where\_c = X[np.where(y == c)]

self.parameters.append([])

for j in range(X.shape[1]):

col = X\_where\_c[:, j]

parameters = {"mean": col.mean(), "var": col.var()}

self.parameters[i].append(parameters)

def \_calculate\_likelihood(self, mean, var, x):

""" Вероятность получить x при нормальном распределении

с матожиданием(mean) среднеквадратичным отклонением(var). """

eps = 1e-4 # Чтобы не было деления на 0.

coeff = 1.0 / np.sqrt(2.0 \* np.pi \* var + eps)

exponent = np.exp(-(((x - mean) \*\* 2) / (2 \* var + eps)))

return coeff \* exponent

def \_calculate\_prior(self, c):

""" Подсчёт априорной вероятности класса c."""

X\_where\_c = self.X[np.where(self.y == c)]

n\_class\_instances = X\_where\_c.shape[0]

n\_total\_instances = self.X.shape[0]

return n\_class\_instances / n\_total\_instances

def \_classify(self, sample):

""" Классификация на основе формулы Байеса: P(Y|X) = P(X|Y)\*P(Y)/P(X)

P(X|Y) - Условная вероятность признаков X при условии класса Y.

рассмиатривается Гауссовское распределение (\_calculate\_likelihood)

P(Y) - априорные вероятности (\_calculate\_prior)

P(X) - Считать не обязательно, т.к. это всего лишь нормировка.

Таким образом, необходимо найти max(P(Y|X))

"""

posteriors = []

for i, c in enumerate(self.classes):

posterior = self.\_calculate\_prior(c)

# Наивное предположение, что все признаки независимы, т.е.

# P(x1,x2,x3|Y) = P(x1|Y)\*P(x2|Y)\*P(x3|Y)

for j, params in enumerate(self.parameters[i]):

sample\_feature = sample[j]

# Вычисление P(x|Y)

likelihood = self.\_calculate\_likelihood(params["mean"], params["var"], sample\_feature)

posterior \*= likelihood

# Суммарная апостериорная вероятность = P(Y)\*P(x1|Y)\*P(x2|Y)\*...\*P(xN|Y)

posteriors.append(posterior)

# Возвращаем класс с наибольшей апостериорной вероятностью.

index\_of\_max = np.argmax(posteriors)

return self.classes[index\_of\_max]

def predict(self, X):

return [self.\_classify(sample) for sample in X]

knn.py:

from collections import Counter

from scipy.spatial.distance import euclidean

import numpy as np

class KNNClassifier:

def \_\_init\_\_(self, k=5, distance\_func=euclidean):

self.X = None

self.y = None

self.k = None if k == 0 else k # Т.к. l[:None] возвращает весь список.

self.distance\_func = distance\_func

def aggregate(self, neighbors\_targets):

most\_common\_label = Counter(neighbors\_targets).most\_common(1)[0][0]

return most\_common\_label

def predict(self, X=None):

predictions = [self.\_predict\_x(x) for x in X]

return np.array(predictions)

def \_predict\_x(self, x):

"""Предсказание метки одного объекта x"""

# рассчёт расстояний между x и всеми остальными объектами из обучающей выборки.

distances = (self.distance\_func(x, example) for example in self.X)

neighbors = sorted(((dist, target)

for (dist, target) in zip(distances, self.y)),

key=lambda x: x[0])

neighbors\_targets = [target for (\_, target) in neighbors[:self.k]]

return self.aggregate(neighbors\_targets)

def fit(self, X, y):

self.X = X

self.y = y

visual.ipynb:

%matplotlib inline

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import RidgeClassifier

from sklearn.metrics import accuracy\_score

from sklearn.preprocessing import StandardScaler

from sklearn.manifold import TSNE

import seaborn as sns

import matplotlib.pylab as plt

import pandas as pd

df = pd.read\_csv('diabetes.csv')

X = df.drop('res', axis=1)

y = df[['res']]

df.head(), df.shape

df.describe()

plot = sns.pairplot(df, hue='res', x\_vars=["Pregnan",

"Gluc",

"Pressure",

"SkinThick",

"Insulin",

"BMI",

"DPF",

"Age",

"res"], y\_vars=["Pregnan",

"Gluc",

"Pressure",

"SkinThick"])

plot.savefig('pairplot1.png')

plot = sns.pairplot(df, hue='res', x\_vars=["Pregnan",

"Gluc",

"Pressure",

"SkinThick",

"Insulin",

"BMI",

"DPF",

"Age",

"res"], y\_vars=["Insulin",

"BMI",

"DPF",

"Age",

"res"])

plot.savefig('pairplot2.png')

fig, axes = plt.subplots(nrows=2, ncols=4, figsize=(18, 10))

features = list(set(df.columns) - set(['res']))

print(features)

for idx, feat in enumerate(features):

sns.boxplot(x='res', y=feat, data=df, ax=axes[idx // 4, idx % 4])

axes[idx // 4, idx % 4].legend()

axes[idx // 4, idx % 4].set\_xlabel('res')

axes[idx // 4, idx % 4].set\_ylabel(feat)

analyze.ipynb:

%matplotlib inline

from collections import defaultdict

from sklearn.model\_selection import train\_test\_split

from sklearn.model\_selection import cross\_validate, learning\_curve, KFold

from sklearn.metrics import accuracy\_score, f1\_score

import pandas as pd

import numpy as np

import knn, bayes

from matplotlib import pylab as plt

import sklearn.svm as svm

from sklearn.tree import DecisionTreeClassifier

df = pd.read\_csv('diabetes.csv')

X = df.drop('Outcome', axis=1)

X = np.array(X)

y = df[['Outcome']]

y = np.array(y)

y = y.ravel()

# KNNClassifier(17)

accuracy = []

x = [0.2, 0.4, 0.6, 0.8, 0.99]

for i in x: # измерение точности при разном количестве входных данных.

temp = []

for \_ in range(100):

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=i)

knn\_clas.fit(X\_test, y\_test)

y\_predict = knn\_clas.predict(X\_test)

temp.append(accuracy\_score(y\_test, y\_predict))

accuracy.append(np.array(temp).mean())

plt.plot(x, accuracy)

plt.xlabel('размер тестовой выборки')

plt.ylabel('точность')

# Naive Bayes Classifier

accuracy = []

kf = KFold(7, random\_state=2, shuffle=True)

for train\_idx, test\_idx in kf.split(X, y): # измерение точности по кросс-валидации.

bayes\_clas.fit(X[train\_idx], y[train\_idx])

y\_predict\_test = bayes\_clas.predict(X[test\_idx])

y\_predict\_train = bayes\_clas.predict(X[train\_idx])

accuracy.append((accuracy\_score(y[train\_idx], y\_predict\_train),

accuracy\_score(y[test\_idx], y\_predict\_test)))

train\_acc = [i[0] for i in accuracy]

test\_acc = [i[1] for i in accuracy]

train\_acc = np.array(train\_acc)

test\_acc = np.array(test\_acc)

print("Accuracy\n train:", round(train\_acc.mean(), 3), ', test:', round(test\_acc.mean(), 3))

plt.plot(train\_acc)

plt.plot(test\_acc)

# Decision tree

fig, axes = plt.subplots(nrows=2, ncols=4, figsize=(18, 10))

print("")

for idx, depth in enumerate([2, 5, 8, 11]):

tree = DecisionTreeClassifier(max\_depth=depth)

accuracy = []

kf = KFold(7, shuffle=True)

for train\_idx, test\_idx in kf.split(X, y): # измерение точности по кросс-валидации.

tree.fit(X[train\_idx], y[train\_idx])

y\_predict\_test = tree.predict(X[test\_idx])

y\_predict\_train = tree.predict(X[train\_idx])

accuracy.append((accuracy\_score(y[train\_idx], y\_predict\_train),

accuracy\_score(y[test\_idx], y\_predict\_test)))

train\_acc = [i[0] for i in accuracy]

test\_acc = [i[1] for i in accuracy]

train\_acc = np.array(train\_acc)

test\_acc = np.array(test\_acc)

print(f'Maxdepth={depth}\n train:', round(train\_acc.mean(),3),', test:', round(test\_acc.mean(), 3))

plt.subplot(220 + idx + 1)

plt.plot(train\_acc)

plt.plot(test\_acc)

#SVM Classifier

svm\_lin\_clas = svm.LinearSVC()

kf = KFold(7, shuffle=True)

accuracy=[]

for train\_idx, test\_idx in kf.split(X, y): # измерение точности по кросс-валидации.

svm\_lin\_clas.fit(X[train\_idx], y[train\_idx])

y\_predict\_test = svm\_lin\_clas.predict(X[test\_idx])

y\_predict\_train = svm\_lin\_clas.predict(X[train\_idx])

accuracy.append((accuracy\_score(y[train\_idx], y\_predict\_train),

accuracy\_score(y[test\_idx], y\_predict\_test)))

train\_acc = [i[0] for i in accuracy]

test\_acc = [i[1] for i in accuracy]

train\_acc = np.array(train\_acc)

test\_acc = np.array(test\_acc)

plt.plot(train\_acc)

plt.plot(test\_acc)

plt.xlabel('номер фолда для тестирования')

plt.ylabel('точность')

plt.title('SVM\_linear')

print("Средняя точность: ", test\_acc.mean())