Министерство образования Республики Беларусь

Учреждение образования Белорусский государственный университет информатики и радиоэлектроники

Факультета компьютерных систем и сетей

Кафедра информатики

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту на тему

Предсказатель переходов на основе решающих деревьев

Выполнил: студент группы 853504

Гаврик А.В.

Руководитель: ассистент кафедры информатики

Леченко А.В.

Минск 2020

Содержание

Введение……………………………………………………………………....3

1. Предсказатели переходов…...…………………………………….............4

* 1. Статическое и динамическое предсказание переходов…….…...4

1.2 Классические алгоритмы предсказателей переходов ……….….5

2. Машинное обучение………………...…………………………………......8

2.1 Основные методы машинного обучения……………..……..........8

2.2 Машинное обучение через решающие деревья …………............9

2.3 Случайный лес………………………………….….……………..10

3. Симмулятор…………………………………...………………………......11

4. Методика работы с полученной программой………………………......12

5. Заключение……………………………………………………………….15

6. Список использованных источников…………………………………...16

**Введение**

Целью курсовой работы является нахождение решения для задачи классификации переходов, используя решающие деревья. Мы рассмотрим статические и динамические предсказатели переходов и их классические алгоритмы. Рассмотрим основные методы машинного обучения. Изучим алгоритм построения дерева решений и ознакомимся со случайным лесом.

**1. Предсказатели переходов.**

В компьютерной архитектуре предсказатель переходов — это устройство, являющиеся частью микропроцессора, которые имеют конвейерную архитектуру. Предсказатель переходов определяет направление ветвлений, что в дальнейшем поможет предсказать осуществиться переход или нет. С помощью предсказателей можно сделать предварительную выборку инструкций, данных из памяти. Еще он позволяет выполнить инструкции, которые находятся после условного перехода, еще до того момента как он будет выполнен.

Основная цель предсказателя переходов это — улучшение потока в конвейере команд. Так как точность предсказания может превышать девяносто процентов он стал присущ для всех современных микропроцессоров, потому что позволяет использовать вычислительные ресурсы процессора наиболее оптимально.

Также предсказатель переходов используется для достижения максимальной эффективности в конвейерных архитектурах процессоров (например процессор x86).

**1.1 Статические и Динамические предсказатели переходов**

Простейшим из методов предсказания переходов является статическое предсказание, потому что оно основывается только на инструкции переходов, а не на информации, полученной в результате динамического выполнения кода. Ранние реализации архитектур RISC, применяли однонаправленное статическое предсказание. Такие предсказания показывают, когда условный переход не выполнится.

Как мы поняли простейшие предсказатели переходов позволяют предсказать какой переход будет выполнен всегда, либо не выполнится.

Динамическое предсказывание собирает информацию о взятых или не взятых ветвях во время выполнения кода. Опираясь на эту информацию, динамический предсказатель прогнозирует результат ветвления. В качестве примера динамического предсказания можно рассмотреть двухуровневый адаптивный алгоритм, который был использован процессорами архитектуры P6. Существовала таблица, в которой содержалась информацию о вероятности условного перехода и сами адреса инструкции. Это таблица анализировалась выдавался результат, и после каждого перехода таблица обновлялась.

**1.2 Классические алгоритмы предсказателей переходов**

Существуют одноуровневые и двухуровневые и многокомпонентные предсказатели переходов. Рассмотрим первые два.

Для одноуровневого предсказания переходов, в случае прямого отображения и просто адреса перехода в качестве источника информации, нужно взять *k* младших битов адреса инструкции перехода, чтобы сопоставить его с записью таблицы с одним столбцом с *2k* записи. Каждая запись обычно состоит из n-битного счетчика, который в большинстве случаев спроектирован как счетчик с насыщением. Счетчик с насыщением работает следующим образом: каждый раз, когда выполняется переход, счетчик увеличивается; если максимальное значение уже достигнуто, ничего не делается. Каждый раз, когда переход не выполняется, счетчик уменьшается; при достижении минимального значения равному ноль, оно остается прежним.

Теперь прогнозирование работает по следующему принципу: если значение счетчика ниже *2 n-1*, ветвь прогнозируется как не занятая, в противном случае ветвь прогнозируется как взятая.

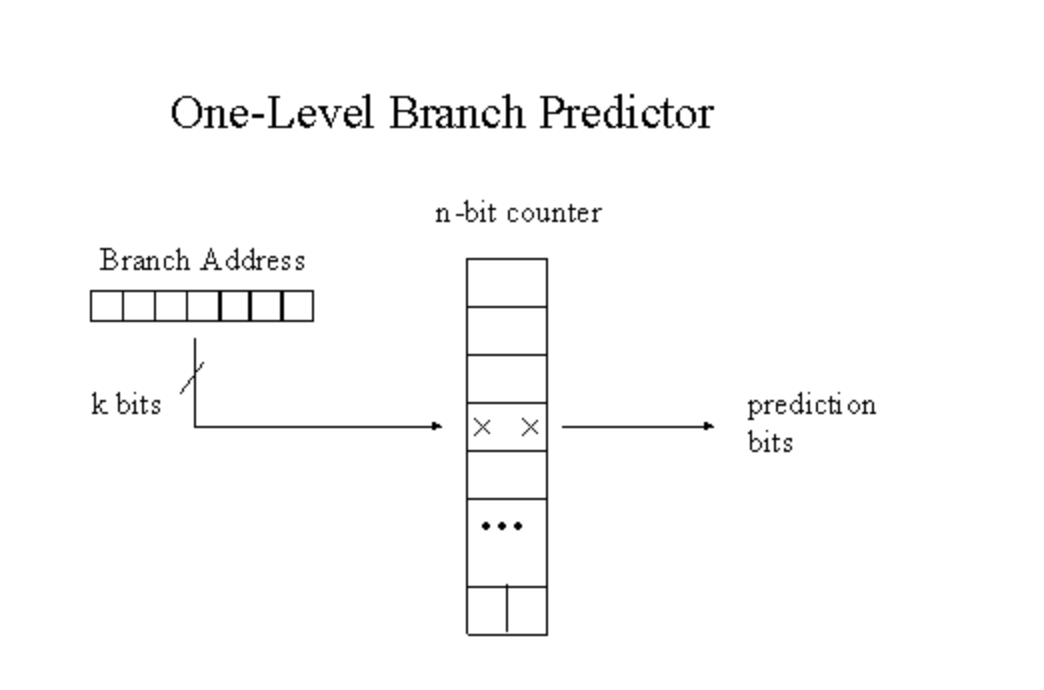


Рисунок 1. Одноуровневый предсказатель переходов.

Двухуровневый предсказатель переходов, использует двумерную таблицу счетчиков. Из-за того, что результат ветки зависят не только от адреса ветки, но также и от результата других недавних ветвей (межветвенная корреляция) и более длительной истории одной и той же ветки. Записи в таблице это — двухбитовые счетчики. Поэтому для того, чтобы предсказать исход ветвления и выбрать запись из таблицы, мы имеем:

1. Branch Адрес. Он использует *k* младших разрядов адреса инструкции перехода.
2. Global Branch History - это сдвиговый регистр, в котором сохраняется результат любого перехода. «Единица» сохраняется для взятой ветви, а «ноль» — для невыбранной. Регистр сдвигается при сохранении самого нового значения. Чтобы обратиться к таблице, рассматриваются *n* последних результатов ветвления.
3. Таблица локальной истории — это таблица регистров сдвига, своего рода глобальная история ветвлений. Однако каждый сдвиговый регистр относится к последним результатам одной отдельной ветви. Поскольку к этой локальной таблице истории обращаются как к одноуровневой таблице предсказания ветвлений, не гарантируется, что не произойдет перекрытия ветвей, и в одном регистре сдвига может храниться информация о разных ветвях.

Поскольку таблица имеет только два измерения, для доступа к строкам и столбцам необходимо выбрать два из трех источников информации.

В целом можно сказать, что двухуровневый предсказатель ветвления более точен, чем одноуровневый предсказатель ветвления, но это преимущество также связано с недостатком более дорогостоящей реализации и тем фактом, что так называемая фаза разогрева , то есть время, в течение которого записи таблицы содержат полезные значения, намного больше.

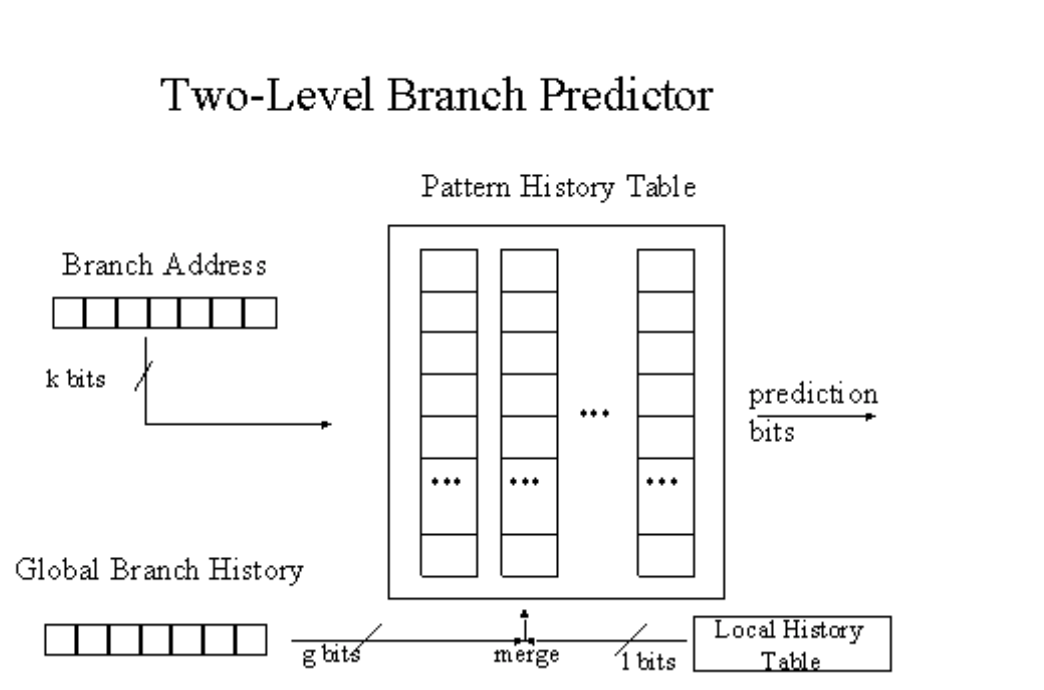


Рисунок 2. Двухуровневый предсказатель переходов.

**2. Машинное обучение**

Машинное обучение — метод анализа данных, который автоматизирует построение аналитических моделей. Это ветвь искусственного интеллекта, основанная на идее, что системы могут учиться на данных, определять закономерности и принимать решения с минимальным вмешательством человека.Существуют два типа обучения. Обучение по прецедентам (также называемое индуктивное обучение), которое находит общие закономерности по частным эмпирическим данным. Второй тип дедуктивное обучение, упорядочивает знания, полученные от экспертов, и переносит их в базу данных.

**2.1 Основные методы машинного обучения**

Машинное обучение с учителем. Пусть на вход поступают переменные x. На выходе мы должны получить переменную y. Будем использовать алгоритм для изучения функции, которая сопоставляет входные данные к выходным Y=f(x). Цель задачи состоит в аппроксимации функции, чтобы при наличии любых входных данных мы могли предсказать выходные данные. Данный метод называется обучение с учителем, потому что процесс обучения данному алгоритму на основе входных данных мы рассматриваем как учителя, который контролирует весь процесс обучения. Обучение можно прекращать, когда алгоритм будет работать с нужным уровнем производительности.

Машинное обучение без учителя. В отличие от обучения с учителем мы не имеем выходных переменных. Нашей задачей является построить модель базовой структуры, чтобы узнать о входных данных. При обучении без учителя машина должна сама исследовать данные и выявлять закономерности. Происходить группировка входных данных в кластеры, которые происходит на основании статистических свойств.

Полу-контролируемые машинное обучение. Происходит тогда, когда у нас есть большой объём входных данных, но мы не имеем весь набор выходных данных. Для решения задачи мы может использовать методы обучения без учителя для обнаружения и изучения структуры данных, поступающих на вход. И можем использовать учителя, для того чтобы делать точные прогнозы для необозначенных данных. Затем обратно передавать данные программе.

**2.2 Дерево решений**

Деревья решений один из важнейших алгоритмов машинного обучения. Они позволяют достичь высокой точности во многих задачах, обладая при этом относительной простотой в реализации. Полученная иерархическая модель легко понятна для интерпретации любым человеком.

Деревья решений подразделяются на деревья классификации и деревья регрессии. Задача деревьев классификации состоит в том, что нужно предсказать отдельный класс, к которому данные принадлежат. В то время как деревья регрессии созданы для предсказания целочисленного результата.

Алгоритмы для создания деревьев решений обычно работают сверху вниз, выбирая переменную на каждом шаге, которая лучше всего разбивает набор значений. Разные алгоритмы используют разные метрики для определения лучшего разбиения. Два наиболее известных алгоритма - прирост информации и неопределенность Джини.

Дерево решений можно строить очень долго, даже до тех пор, пока оно не будет иметь тысячи веток. Поэтому существуют различные критерии остановки. Самый известный из этих критериев - использование минимального счетчика для каждого узла. Если этот счетчик меньше, чем определенное минимальное значение, то разбиение невозможно и узел становится конечным листом. Когда все наши узлы становятся конечными, тренировка дерева прекращается. Таким образом, если счетчик будет иметь совсем небольшое значение, разбиения будут более точными и более информативными, но мы рискуем получить переобученное дерево. Или, наоборот, имея большое значение счетчика, наше дерево может оказаться совсем не полезным для анализа. Выбор минимального значения счетчика зависит от набора данных, с которым приходится работать.

**2.3 Случайный лес**

Деревья решений могут выдавать нестабильные результаты для определенных данных для обучения. Это происходит из-за того, что в деревьях решения бывает высокая дисперсия. Для решения этой проблемы существует алгоритм случайный лес.

Случайны лес не только строит деревья по нескольким выборкам обучающих данных, но также в ходе построения деревьев он ограничивает функции, которые можно для этого использовать. Что в результате приводит к тому, что деревья различаются между собой. Это помогает повысить производительность. Данный алгоритм является универсальным и быстро обучаемым для того, чтобы найти связи внутри набора данных.



Рисунок 3: Диаграмма случайного леса

# **3. Симулятор RISC-V Simulator**

В данной курсовой работе мы используем RISC-V Simulator (by hehao98).

RISC-V Simulator — простой симулятор ЦП RISC-V с 5-этапным конвейером, прогнозированием переходов и моделированием кэша. Поддерживает набор инструкций RV64I в пользовательском режиме.

**4. Методика работы с полученной программой**

После установки RISCV-Simulator нам необходимо записывать данные по branch predictors, в csv файл. Для этого необходимо расширить функционал файла simulator.cpp.

****

Рисунок 4. Добавление логирования.

После данной модификации при компиляции, необходимые нам данные будут записаны в csv файл.

Компилируем.



Рисунок 5. Компиляция симулятора.

Выполняем файл matrixmulti.riscv.



Рисунок 6. Выполняем файл.

В результате получаем набор данных(data\_2.csv). Аналогичным образом создаем еще два набора данных.

Затем строим деревья (рассмотрим 1 dataset, остальные выполняются аналогично). Разделяем наш набор данных на тестовые данные и данные для тренировки в соотношении 20:80.

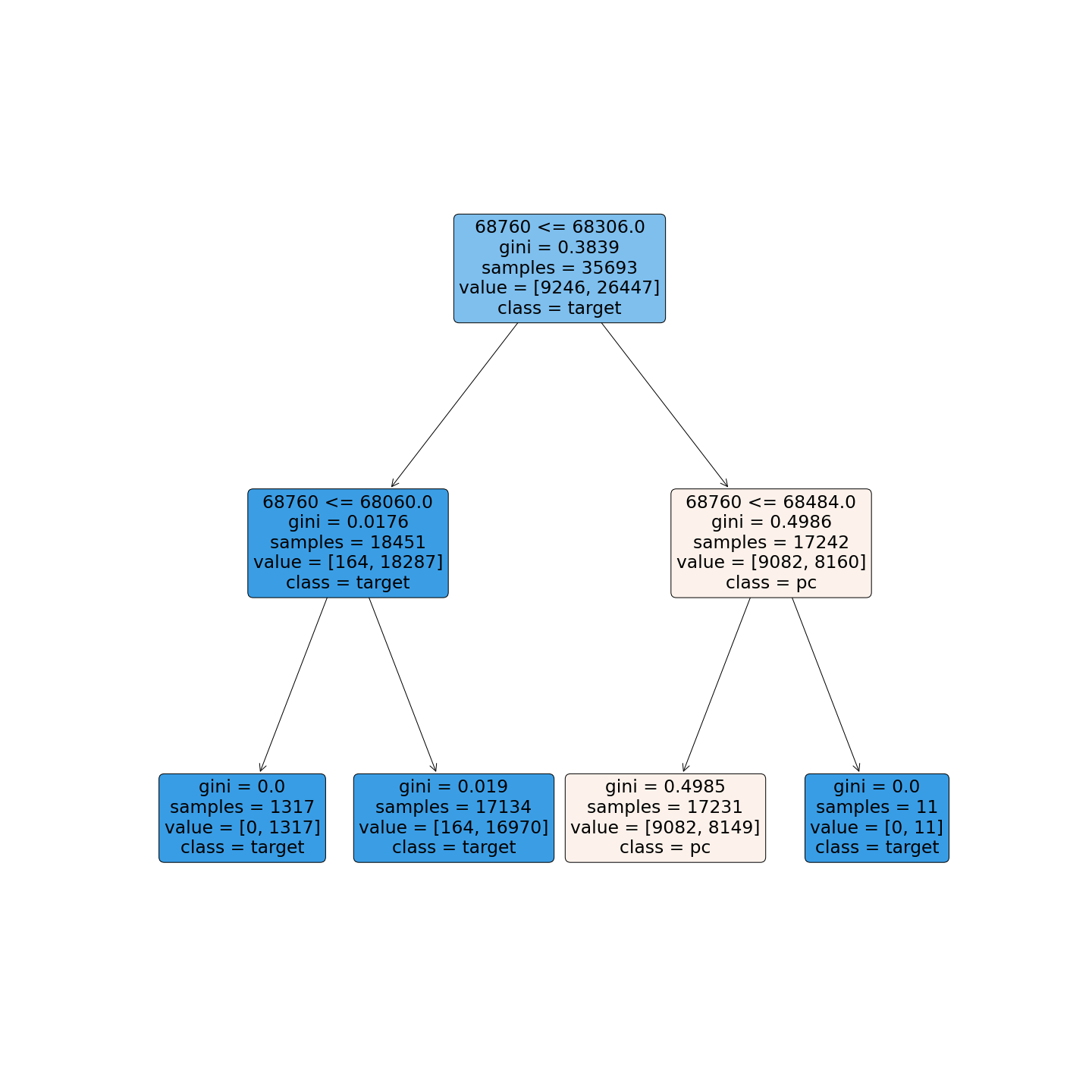


Рисунок 7. Диаграмма дерева решений.

Точность:

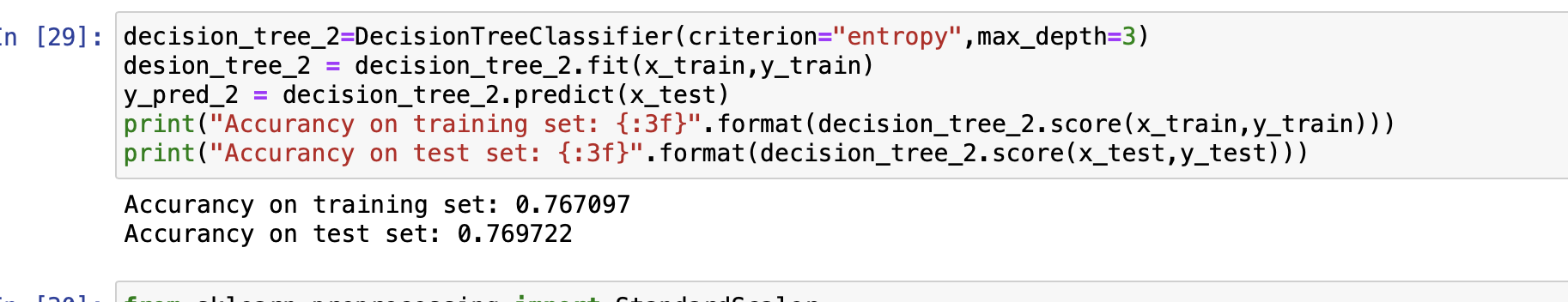


Рисунок 8. Точность предсказания дерева решений на тренировочных и тестовых данных.

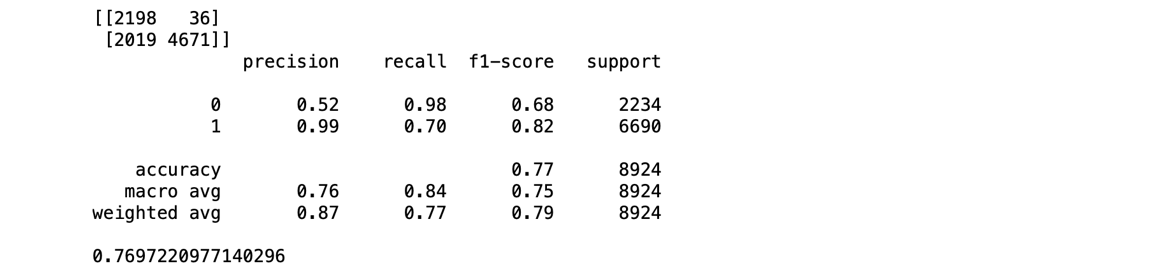


Рисунок 9. Точность, полученная с помощью алгоритма случайного леса, на тренировочных и тестовых данных.

**Заключение**

В результате курсовой работы мы разобрались в понятиях:

Предсказатель переходов, рассмотрели статические и динамические предсказатели переходов и их классические алгоритмы. Разобрались с

Основными методами машинного обучения и научились строить дерево решений для нахождения решения задачи классификации.

**Список использованных источников**

1. <https://vc.ru/life/152868-chto-takoe-derevo-resheniy-i-gde-ego-ispolzuyut>
2. <https://habr.com/ru/company/ods/blog/324402/>
3. <https://www.osp.ru/cio/2018/05/13054535>
4. <http://www.machinelearning.ru/wiki/index.php?title=Машинное_обучение>
5. [https://ru.qaz.wiki/wiki/Branch\_predictor - Static\_branch\_prediction](https://ru.qaz.wiki/wiki/Branch_predictor#Static_branch_prediction)
6. [https://studfile.net/preview/7514499/page:4/ - 13](https://studfile.net/preview/7514499/page:4/#13)
7. <https://ru.wikipedia.org/wiki/MIPS_(архитектура)>
8. <https://ru.qaz.wiki/wiki/Branch_predictor#One-level_branch_prediction>
9. <https://en.wikipedia.org/wiki/Random_forest>
10. https://github.com/hehao98/RISCV-Simulator