средствами — утилитой pkexec. Запущенный таким образом демон осуществляет посылку с результатом подключения на графическую часть, что при удачном подключении позволяет начать процесс обмена.

Данное нестандартное решение позволяет обеспечить выполнение поставленной задачи стандартными средствами при отсутствии настроенного процесса обмена данными по USB на устройстве с ОС Аврора.

Список использованных источников

1. Портал разработчиков АврораОС [Электронный ресурс] URL: https://developer.auroraos.ru/ (Дата обращения: 10.04.2025)

2. Документация по libusb [Электронный ресурс] URL: https://libusb.info/ (Дата обращения: 10.04.2025)

3. Документация Qt Group [Электронный ресурс] URL: https://doc.qt.io/ (Дата обращения: 10.04.2025)

4. Снейдер, Й. Эффективное программирование TCP/IP Й. Снейдер; Пер. с англ. А. Слинкин. - СПб. и др.: Питер, 2001. - 319 с. ил

Жукова Юлия Сергеевна, студент каф. РЭС, инженер АС и ПО ООО «КВАДРО КОД», yulia@4code.ru

УДК 004.852:621.391

ПОДГОТОВКА ОБУЧАЮЩИХ ДАННЫХ ДЛЯ ЗАДАЧ КЛАССИФИКАЦИИ ЧАСТОТНЫХ СИГНАЛОВ И РАЗРАБОТКА МЕТОДА ОПРЕДЕЛЕНИЯ ПОКАЗАТЕЛЕЙ МАТРИЦЫ ОШИБОК КЛАССИФИКАЦИИ

И.М. Берестов, М.Д. Лучин
ООО «КВАДРО КОД», г. Самара

С повсеместным внедрением новых видов летательных аппаратов, а именно БПЛА, появилась потребность классифицировать частотные сигналы, обнаруженные в определенных диапазонах и оценивать качество классификации [1]. В связи с этим была поставлена следующая цель: создать обучающую и тестовую выборку для модели машинного обучения, предназначенной для анализа заданного спектра частот и определения типа сигнала, обеспечив высокое качество данных и минимизацию шумов и разработать метод для подсчета показателей матрицы ошибок классификации.

Для достижения поставленной цели была проделана следующая работа: в целях обучения модели, определяющей тип входящего сигнала, была записана обучающая выборка данных, состоящая из 7 различных типов сигналов, а именно: аналоговое радио, dmr, dig, fpv, gsm, WI-FI и работа РЭБ. Записаны данные в виде набора бинарных файлов формата .c16.

Запись производилась в лабораторных условиях, всего было создано 840 файлов, общим объемом 34,4 ГБ. Из них примерно половина - тестовые, размером по 7,62 МБ на файл и вторая половина - обучающие размером по 76,2 МБ на файл.

Далее был произведен анализ параметров релевантности. За релевантные файлы были приняты те, которые не являются пустыми и не содержат испорченную запись информации, другими словами - «битые». Для того, чтобы определить, насколько файл испорчен - было предположено, что необходимо подсчитать количество подряд идущих нулевых пар байтов. И составить список файлов, упорядоченный по количеству таких пар. Для этой задачи был написан алгоритм на языке C++, который перебирал все файлы из набора данных и составлял из них нужный нам порядок. Таким образом, мы получили список, где вверху были релевантные файлы, а внизу нерелевантные. Однако после перепроверки этого списка, оказалось, что данный метод оценки неэффективен, так как данные представляют собой запись разных типов сигналов, из-за чего файлы могли считаться некорректными, даже если таковыми не являлись. Происходило это из-за того что некоторые типы сигналов, как, например WI-FI, не транслируют данные постоянно, а делают это периодически, с временными промежутками между передачами, как продемонстрировано на рисунке 1, из-за чего эти промежутки, засчитывались как незначащие участки файла.

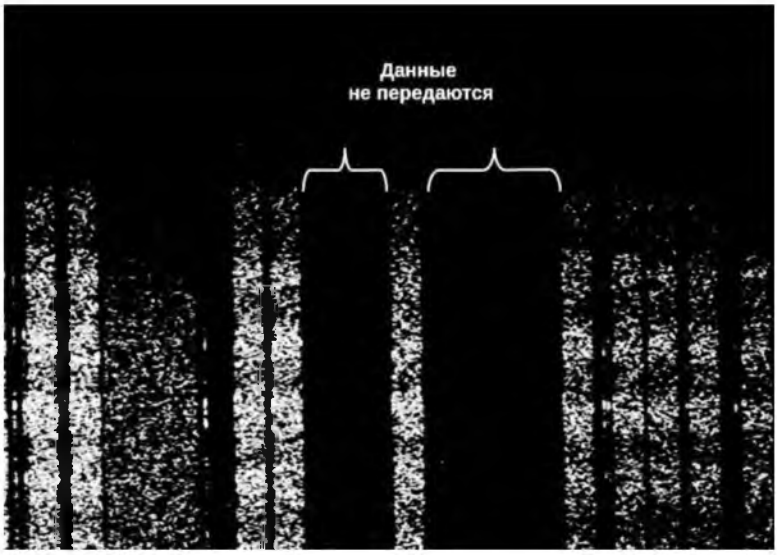

Рисунок 1 – Пример визуализации релевантного файла, засчитанного как нерелевантный

По этой причине был переопределен параметр релевантности. За релевантный файл принимался тот, который содержит в себе более n% ненулевых пар байтов. После этого составленный ранее алгоритм был переделан так, чтобы считать отношение ненулевых пар байтов к нулевым, и составлял список файлов, отсортированный по убыванию заполненности ненулевыми данными. Методом подбора был вычислен коэффициент n и все файлы с коэффициентом ниже n были исключены из выборки.

Для подсчёта показателей матрицы ошибок [2], на языке Java был разработан метод класса, содержащего результаты классификации частотных сигналов по всему определяемому спектру частот.

Метод «getMetrics()» принимает в качестве входных параметров:

1. Тестовую выборку <Частота, тип сигнала>.
2. Переменную типа «bolean», определяющую первую букву метрики, для удобства назовём эту переменную «fOrT» (таблица 1).
3. Переменную типа «bolean», определяющую вторую букву метрики, для удобства назовём эту переменную «nOrP» (таблица 1).
4. Объект перечисления SigType (тип сигнала), для которого рассчитывается данный показатель.
5. Переменную типа «java.util.Date», задающую время начала анализируемого временного интервала.
6. Переменную типа «java.util.Date», задающую время конца анализируемого временного интервала.

Таблица 1 – Входные значения переменных «fOrT» и «nOrP», определяющих конкретный показатель матрицы ошибок.

Наименование	Значение «fOrT»	Значение «nOrP»
FN	false	false
FP	false	true
TN	true	false
TP	true	true

Было решено реализовать алгоритм с помощью встроенных в Java инструментов, а именно пакета «java.util.stream». Алгоритм реализован с использованием методов «filter()», «count()» и разработанных логических выражений.

Основная работа заключалась в оптимизации логики работы алгоритма, необходимо было избегать дублирования кода и нагромождения условных операторов. Важно уточнить, что результаты классификации могут содержать повторы, например, если спектр частот отсканирован и результаты сканирования классифицированы многократно.

Алгоритм работы метода:

1. Полученные результаты классификации фильтруются по времени («filter()»).

2. Полученные результаты классификации фильтруются по наличию частот в тестовой выборке («filter()»).

3. В зависимости от значения переменных «fOrT» и «nOrP» отбираются те классифицируемые объекты, для которых предсказанный параметр либо ложноотрицательный, либо ложноположительный, либо истинно отрицательный, либо истинно положительный («filter()»).

4. Количество нужных результатов классификации подсчитывается при помощи метода «count()».

5. Вычисляется количество результатов классификации, частоты которых соответствуют частотам в тестовой выборке, для нормирования значения («filter()»).

6. Возвращается вычисленное и нормированное значение.

Для проверки работоспособности метода:

1. Был реализован тестовый класс, описывающий структуру <Входные данные, Ожидаемый результат>.

2. Был создан и заполнен необходимыми данными объект тестового класса.

3. При помощи библиотеки тестирования «org.junit.jupiter.api», аннотации «@Test», метода «assertEquals(Ожидаемый результат, полученный результат)», реализована проверка значений, возвращаемых функцией «getMetrics()», в зависимости от поданных в качестве аргументов, содержащихся в объекте тестового класса входных данных.

В результате проведенной работы мы получили выборку из данных, соответствующих требованиям к обучению модели, и работоспособный метод расчета показателей матрицы ошибок для каждого из классифицируемых классов.

Список использованных источников

1. С.А. Князьков, Д.В. Лучин, А.А. Минаев, А.П. Трофимов, В.В. Юдин, Программно-аппаратные средства тестирования обнаружителей беспилотных летательных аппаратов // Электросвязь – 2025. – № 4, с. 25-31, DOI: 10.34832/ELSV.2025.66.4.003

2. Rainio O., Teuho J., Klén R. Evaluation metrics and statistical tests for machine learning // Scientific Reports. – 2024. – Vol. 14. – Article number 6086. – DOI: 10.1038/s41598-024-56706-x.

Лучин Михаил Дмитриевич, студент гр. 6206-010302D, техник ООО «КВАДРО КОД», luchin.md@4code.ru

Берестов Иван Михайлович, студент гр. 6206-010302D, техник ООО «КВАДРО КОД», ivantcharkin@yandex.ru