Министерство науки и высшего образования РФ   
Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский технологический университет «МИСиС»

Институт ИТКН

Кафедра АСУ

**КУРСОВАЯ НАУЧНО-ИССЛЕДОВАТЕЛЬСКАЯ РАБОТА**на тему: «Разработка модели идентификации целенаправленной физической активности человека при выполнении производственных работ»

**Выполнил:**студент гр. БИВТ-18-4

Кириллов П.С..

**Проверил:**

к.т.н., доцент   
Трофимов В.Б.

Москва, 2022

**СОДЕРЖАНИЕ**

[1 Разработка информационного обеспечения 3](#_Toc102839239)

[2 Разработка математического обеспечения 13](#_Toc102839240)

[2.1 Метрики качества алгоритмов: 13](#_Toc102839241)

[2.1.1 Accuracy 13](#_Toc102839242)

[2.1.2 Precision 13](#_Toc102839243)

[2.1.3 Recall 13](#_Toc102839244)

[2.1.4 F1-мера 14](#_Toc102839245)

[2.2 Статистические методы анализа и обработки данных: 15](#_Toc102839246)

[2.2.1 Выборочное математическое ожидание 15](#_Toc102839247)

[2.2.2 Оценка стандартного отклонения (выборочное отклонение) 15](#_Toc102839248)

[2.2.3 Коэффициент асимметрии: 15](#_Toc102839249)

[2.2.4 Коэффициент эксцесса 16](#_Toc102839250)

[2.2.5 Энтропия 16](#_Toc102839251)

[2.2.6 Квантили 16](#_Toc102839252)

[2.2.7 Мода 17](#_Toc102839253)

[2.2.8 QQ-plot 17](#_Toc102839254)

[2.2.9 Обучающая, валидационная, тестовая выборки 18](#_Toc102839255)

[2.2.10 Кодирование зависимого столбца 18](#_Toc102839256)

[2.2.11 Метод фиксированного окна: 19](#_Toc102839257)

[2.3 Алгоритмы машинного обучения: 20](#_Toc102839258)

[2.3.1 Алгоритм градиентного бустинга CatBoost 20](#_Toc102839259)

[2.3.2 Метод опорных векторов (SVM) 21](#_Toc102839260)

[2.3.3 Алгоритм Random Forest (случайный лес) 23](#_Toc102839261)

[3 Разработка программного обеспечения 28](#_Toc102839262)

[4 Компьютерное моделирование 35](#_Toc102839263)

[5 Анализ результатов компьютерного моделирования 43](#_Toc102839264)

## **1 Разработка информационного обеспечения**

В качестве задачи ставится разработка программного решения, позволяющего распознавать рабочую активность рабочих, занятых физическим трудом. В качестве исследуемых данных была использована телеметрия, считанная с умных часов. Данные (датасеты) представлены в виде таблиц csv формата, где строки - наблюдения за сотрудниками в разные моменты времени, а столбцы - признаки, характеризующие рабочего уровня исполнения по определенным свойствам.

Таблица 1 - Признаки в таблицах телеметрии сотрудников.

|  |  |
| --- | --- |
| Название признака | Описание признака |
| 'Дата' | Время получения данных с устройства |
| 'Дата\_получения\_локации' | Время получения данных со спутника |
| 'Широта' | Географическая координата |
| 'Долгота' | Географическая координата |
| 'Высота' | Географическая координата |
| 'Скорость' | Физическая скорость перемещения рабочего, считана посредством геолокации умных часов |
| 'Точность' | Точность измерений |
| 'Ax' | Ускорение по оси x |
| 'Ay' | Ускорение по оси y |
| 'Az' | Ускорение по оси z |
| 'Сердечный\_Ритм' | Ритм сердца человека |
| 'Шаги' | Количество шагов, совершенное человеком |

Продолжение таблицы 1

|  |  |
| --- | --- |
| Название признака | Описание признака |
| 'Разметка' | Метка класса, столбец, подлежащий распознаванию по признакам выше |

Отдельно стоит отметить, что признак “Шаги” был получен c одинаковыми интервалами с устройства мониторинга. Так как данные вдоль строк - временной интервал, то шаги накапливались со временем. По этой причине признак в настоящий момент не является информативным и должен быть обработан.

Был выбран класс алгоритмов обучения с учителем (supervised learning) для классификации, так как он позволяет однозначно определять качество выбранных моделей машинного обучения путем сравнения предсказанных меток зависимого признака с существующей разметкой. Изначальная разметка считываемых данных была следующей:

Таблица 2 - Изначальная кодировка меток классов.

|  |  |
| --- | --- |
| Наименование | Кодировка |
| Простой (стоит на месте) | a |
| Передвижение по территории | b |
| Вкручивание/выкручивание болтов/шурупов | с |
| Перенос крупных предметов по территории | d |
| Поднятие предмета | e |
| Положить предмет | f |
| Перенос мелких предметов по территории | g |
| Подключение/отключение проводов от компьютера | h |

Продолжение таблицы 2

|  |  |
| --- | --- |
| Наименование | Кодировка |
| Протирание предметов от пыли | i |
| Упаковке предметов в коробки | j |
| Переписка или звонок по телефону | k |
| Невозможно определить | l |

Данные метки являются слишком узконаправленными и не подходят для унифицированной классификации целенаправленной физической деятельности рабочих, поэтому было решено перекодировать их для определения видов активности в общем виде.

Таблица 3 - Перекодирование меток классов (группировка по активности).

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование (старое) | Кодировка (старая) | Кодировка (новая) | Наименование (новое) |
| Простой (стоит на месте) | a | a | Простой |
| Передвижение по территории | b | b | Перемещение |
| Вкручивание/выкручивание болтов/шурупов | c | e | Низкая мобильная активность |
| Перенос крупных предметов по территории | d | c | Низкая статическая активность |
| Поднятие предмета | e | e | Низкая статическая активность |
| Положить предмет | f | e | Низкая статическая активность |
| Перенос мелких предметов по территории | g | d | Высокая мобильная активность |
| Подключение/отключение проводов от компьютера | h | f | Высокая статическая активность |

Продолжение таблицы 3

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование (старое) | Кодировка (старая) | Кодировка (новая) | Наименование (новое) |
| Протирание предметов от пыли | i | d | Высокая мобильная активность |
| Упаковка предметов в коробки | j | c | Низкая мобильная активность |
| Переписка или звонок по телефону | k | a | Простой |
| невозможно определить | l | При анализе не рассматривать | |

Подробное описание новых меток классов представлено в таблице 10.

Таблица 4 - Значение новых меток классификации.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № п.п. | Наименование | Характеристика | Критерий определения | Кодировка |
| 1 | Простой | Отсутствие каких-либо действий | Фиксация отсутствия интенсивных движений руками и перемещения в пространстве за интервал времени | a |
| 2 | Перемещение | Простое перемещение в пространстве | Фиксация перемещения в пространстве со средней интенсивностью движений руками | b |

Продолжение таблицы 4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № п.п. | Наименование | Характеристика | Критерий определения | Кодировка |
| 3 | Низкая мобильная активность | Перемещение в пространстве с выполнением простых (низкоамплитудных) действий ручного труда | Фиксация перемещения в пространстве с низкой интенсивностью или отсутствием движений руками | c |
| 4 | Высокая  мобильная активность | Перемещение в пространстве с выполнением сложных (высокоамплитудных) действий ручного труда | Фиксация перемещения в пространстве с высокой интенсивностью движений руками | d |
| 5 | Низкая статическая активность | Выполнение простых (низкоамплитудных) действий ручного труда без перемещения в пространстве | Фиксация движений руками со средней интенсивностью и отсутствие перемещений в пространстве | e |
| 6 | Высокая статическая активность | Выполнение сложных (высокоамплитудных) действий ручного труда без перемещения в пространстве | Фиксация движений руками с высокой интенсивностью и отсутствие перемещений в пространстве | f |

Для анализа данных в научных работах с похожей задачей телеметрия рабочих визуализируются посредством гистограмм, boxplot диаграмм и вывода статистической информации, такие же методы будут использованы в решаемой задаче.

Для анализируемых данных актуальна проблема накопления количества шагов со временем. По этой причине признак не является информативным, для решения этой проблемы нужно вычесть из записей признака “Шаги” количество шагов, которые были сделаны до последнего акта записи данных. После чего колонки предикторов нормализуются.

Обработка данных во всех работах со схожей задачей включает в себя выделение новых признаков при помощи извлечения статистических свойств из исходных колонок таблиц. Количество выделяемых статистических свойств в НТИ различается, поэтому выделим список самых частых признаков, которые будут способны охватить большинство полезной информации для классификации: 25 и 75 квантили, математическое ожидание, среднеквадратичное отклонение, коэффициент асимметрии, эксцесса минимум, максимум и энтропия. Решено было использовать непересекающиеся интервалы агрегации данных, как это сделано в большинстве рассмотренных работ. Помимо избавления от шумов и выделения полезной информации агрегация должна хорошо работать после интеграции разрабатываемого решения, где данных гораздо больше, чем в текущей работе. Для обработки зависимого признака используется нахождение моды в интервалах агрегации и перезапись изначальной метки полученной величиной. Для обоих видов обработки признаков размер окна выбран длиной в 5 строк данных. Для моделей, которые требуют кодирования зависимого столбца числами будет использован Label Encoder.

Статистическая обработка выбросов на основе гипотез нормальности не используется в задачах распознавания целенаправленной физической активности в НТИ и в данной применяться не будет, поскольку при удалении данных теряется потенциально полезная информация, включая метки классов. Так как данные размещались вручную, то потеря такой информации нежелательна. Однако, будут удаляться признаки, которые бессмысленны для прогноза (например, местонахождение сотрудников), также удаляются признаки, значения в которых не содержат в себе полезной информации (например, признак содержит одно уникальное значение - 0).

После вышеобозначенной обработки признаков две таблицы данных объединяются в одну для дальнейшего ее задействования в задаче отбора модели.

После этих средств используется разбиение на обучающую, валидационную и тестовую выборки (в пропорция разбиения 70%, 15%, 15% соответственно). Обучающая подвыборка используются для настройки моделей, валидационная нужна для их сравнения на метриках качества, а итоговая мощность выбранной модели рассчитывается на основе на тестовой. Кросс-валидация не будет использоваться, поскольку ее работа требует существенных временных затрат, в отличии от рассмотренного выше способа.

Как было показано в обзоре НТИ, наилучшими алгоритмами в среднем для задачи классификации целенаправленной физической активности являются: SVM, Random Forest. Несмотря на проявленную эффективность SVM, он имеет существенные минусы, которые мешают его использованию в решаемой задачи, а именно:

* сильная неустойчивость к аномалиям в данных;
* сложность выбора спрямляющего пространства для признаков (в т.ч. выбора ядра) и настройки гиперпараметров для увеличения мощности алгоритма;
* свойственно переобучение в виду принципов работы алгоритма;

Так как рассматриваемые данные, ввиду способа их получения, имеют много выбросов, то для качественной работы метода опорных векторов эти аномалии нужно будет удалять, а это может плохо сказаться на полезности датасетов для обучения модели. Алгоритм Random Forest не имеет вышеописанных недостатков, он переобучается гораздо меньше, крайне эффективен при гиперпараметрах по умолчанию и не чувствителен к аномалиям в данных. К сожалению, при изменении гиперпараметров существенного увеличения мощности модели не произойдет ввиду особенностей работы алгоритма.

Алгоритм CatBoost, основанный на градиентном бустинге, является потенциально мощнейшим из рассматриваемого списка, так как не имеет проблем, актуальных для SVM и RF (Random Forest) и обладает потенциально большей предсказательной силой. Стоит отметить, что научных работ по теме классификации целенаправленной физической активности рабочих этим алгоритмом нет, однако, он довольно новый и вполне вероятно, что в ближайшем будущем такие работы появятся. Для рассмотрения были выбраны алгоритмы Random Forest и CatBoost Classifier с целью выявить наилучший.

Для сравнения моделей и несмещенной оценки выбранной использованы те метрики, которые встречаются в научных работах наиболее часто: accuracy, precision, recall, confusion matrix, f1-score.

Вывод статистической результирующей классификационной информации будет осуществлен посредством использования гистограммы распределения данных по предсказанным меткам.

Рассмотрим модель построения решения задачи. В качестве входных потоков выступают два датасета с телеметрией рабочих.

Закодируем входные потоки в модель следующим образом:

- i-ый датасет со всеми признаками.

Верхний индекс датасета означает первую букву последней совершенной над ним операции. Например, обозначение означает 1-ый датасет, обозначение последней операции над которым начиналось с буквы F

Таблица 5 - Операции, совершаемые над входными данными.

|  |  |  |  |
| --- | --- | --- | --- |
| Обозначение операции | Входные параметры | Выходные | Действие |
| Analysis | , | u\_feature (бесполезные признаки) | Проверяет входные данные на наличие неинформативных признаков |
| Format | ,,u\_feature | , | Удаляет неинформативные признаки из наборов данных, обрабатывает признак “Шаги” и преобразует метки классов в соответствии с схемой \_\_ |
| Normalize | , | , | Нормализует столбцы датасетов с целью приведения данных к общему масштабу |

Продолжение таблицы 5.

|  |  |  |  |
| --- | --- | --- | --- |
| Обозначение операции | Входные параметры | Выходные | Действие |
| Gen Features | , | , | Генерация новых признаков из исходных и обработка зависимого столбца |
| Agg | , |  | Два датасета объединяются в один |
| Train Test Split |  |  | Разбиение исходных данных на 3 подвыборки: 1 - обучающая, 2 - валидационная, 3 - тестовая |
| CatBoost Learning Check |  | - результаты классификации моделью тестовой выборки на выбранных метриках, - предсказанные метки классов, - модель catboost | Происходит обучение и тестирование модели CatBoost классификатора |
| Label Encoder |  |  | Кодирует строковые метки классов числами |
| Random Forest Learning Check |  | , , | Происходит обучение и тестирование модели случайного леса |

Продолжение таблицы 5.

|  |  |  |  |
| --- | --- | --- | --- |
| Обозначение операции | Входные параметры | Выходные | Действие |
| Compare Models | , | (название лучшего алгоритма на метриках) | Сравнение выбранных алгоритмов на основе результатов обучения |
| Best Model Final Check | , , , | , ,  (результаты классификации выбранной моделью) | Если - CatBoost, то - модель CatBoost, иначе - Random Forest. Аналогично для . Операция осуществляет несмещенную оценку модели на тестовой выборке для определения ожидаемой мощности. |
| Generate Predict Visualization |  |  | Генерирует визуализацию распределения классов для лучшей интерпретации результатов |

## **2 Разработка математического обеспечения**

### **Метрики качества алгоритмов:**

В данной пункте приведены наиболее важные и используемые метрики качества в задачах классификации.

#### **2.1.1 Accuracy**

,(1)

где - алгоритм машинного обучения,

- предсказание на i-ом объекте данных,

- истинное значение на i-ом объекте,

[выражение] - скобки Айверсона (число 0, если условие ложно и 1, если истинно, - длина выборки)

#### **2.1.2 Precision**

,(2)

где TP - верно классифицируемые как положительный класс,

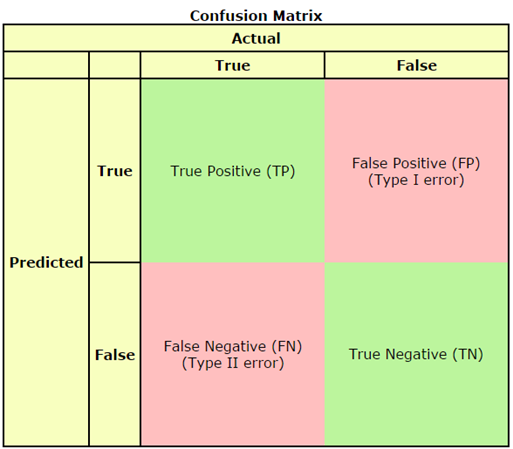
FP - ложно классифицируемые как положительный класс

#### **2.1.3 Recall**

, (3)

где TP - верно классифицируемые как положительный класс, FN - ложно классифицируемые как отрицательный класс.

Рассмотрим матрицу ошибок, в которой TN - правильно классифицируемый отрицательный класс.

  
Рисунок 2 - Пример матрицы ошибок.

#### **2.1.4 F1-мера**

F1-мера является гармоническим средним между точностью и полнотой:

(4)

В случае многоклассовой классификации матрица ошибок расширяется на необходимое количество меток. Информация о точности, полноте и F1-мере дается по каждому прогнозируемому классу отдельно.

### **Статистические методы анализа и обработки данных:**

#### **2.2.1 Выборочное математическое ожидание**

Математическое ожидание будет находиться по выборке данных, а значит имеет следующую формулу:

, (5)

где – элемент выборки, n – ее длина. Часто формула отображается в виде оператора .

#### **2.2.2 Оценка стандартного отклонения (выборочное отклонение)**

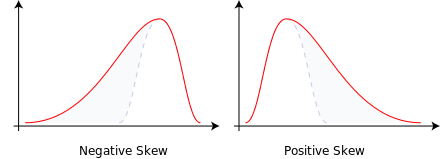
Вычисление оценки стандартного отклонения основано на вычислении выборочной дисперсии (смещенная оценка дисперсии) и вычисляется по следующей формуле:

(6)

#### **2.2.3 Коэффициент асимметрии:**

Величина, которая характеризует асимметрию распределения выборки данных относительно нормального, мера скошенности распределения влево или вправо.

(7)

  
Рисунок 4 - Поведение коэффициента асимметрии.

#### **2.2.4 Коэффициент эксцесса**

Определяет меру остроты пика выборочного распределения относительно нормального распределения. Мера скошенности по вертикали.

(8)

#### **2.2.5 Энтропия**

Энтропия характеризует меру неопределенности в данных.

, (9)

где рассчитывается как:

(10)

#### **2.2.6 Квантили**

, - квантиль (квантиль порядка ) случайной величины , если

(11)

, (12)

где оператор P(condition) обозначает вероятность события, заключенного в скобках

значение

#### **2.2.7 Мода**

Обозначает значение во множестве наблюдений, которое встретилось чаще остальных.

#### **2.2.8 QQ-plot**

Визуальный способ предположений о проверке нормальности выборочных данных.

Этапы построения:

1. Отсортировать выборку по неубыванию
2. Сопоставить каждому объекту выборку точку на графике:
   1. Значение вертикальной оси - значение элемента выборки
   2. Значение по горизонтальной оси - математическое ожидание квантиля стандартного нормального распределения, подсчитанной по выборке такого же объема.

Если выборка взята из нормального распределения, то точки на графике должны лежать на одной прямой:

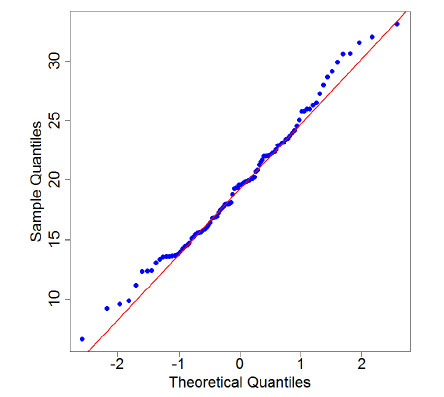


Рисунок 5 - Пример Q-Q графика

#### **2.2.9 Обучающая, валидационная, тестовая выборки**

Данные разделяются на 3 подвыборки: обучающую (большая часть данных 70%), валидационную (15 %) и тестовую (15 %).

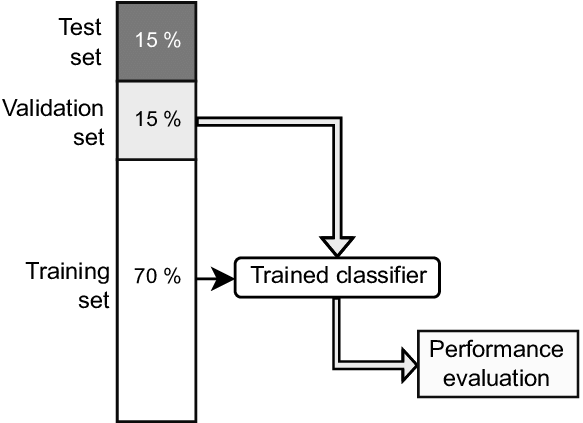


Рисунок 7 - Пример использования разбиения

Также выделяют стратифицированную выборку, в которой пропорции между распознаваемыми классами сохраняется в трех видах подвыборок. Такие разбиения могут использоваться для подбора гиперпараметров модели или для сравнения модели с целью выбрать лучшую из набора.

#### **2.2.10 Кодирование зависимого столбца**

Требуется для кодирования категориальных столбцов числами для подачи в некоторые модели машинного обучения, например, SVM (метод опорных векторов) и Random Forest. Этот метод кодирования обозначает словосочетанием Label Encoder.

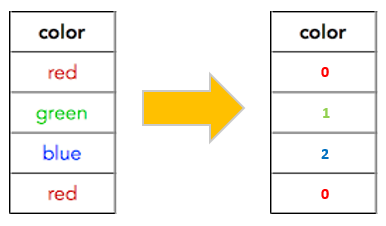
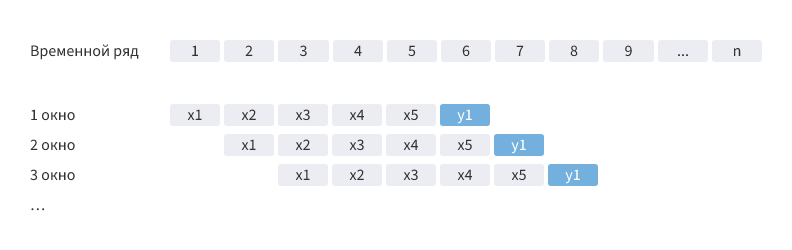


Рисунок 9 - Label Encoder. Пример работы.

Для классификатора CatBoost данный алгоритм не нужен, так как он уже реализован внутри модели.

#### **2.2.11 Метод фиксированного окна:**

Для того, чтобы преобразовать данные с целью извлечения новых признаков или сглаживания аномалий используется метод скользящего окна. Его преимущество заключается в том, что при такой обработке практически не теряется полезной информации.

  
Рисунок 8 - Метод скользящего окна.

Метод скользящего окна использует различные функции, позволяющие извлечь из данных нужную информацию.

Метод фиксированного окна (non-overlapping) отличается лишь разбиением окон на непересекающиеся интервалы

### **Алгоритмы машинного обучения:**

#### **2.3.1 Алгоритм градиентного бустинга CatBoost**

Catboost классификатор - метод градиентного бустинга, реализованный над симметричными деревьями.

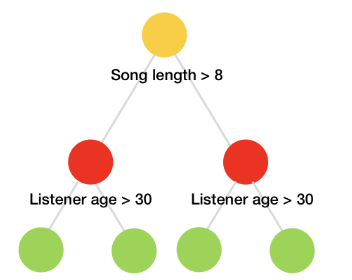
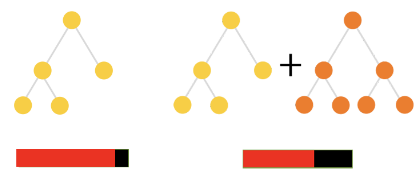


Рисунок 9 - Пример симметричного дерева.

Градиентный бустинг - алгоритм построения композиции простых деревьев, где каждое следующее дерево в композиции исправляет ошибку предыдущих, двигаясь в сторону, обратную градиенту целевой функции ошибки на данных. Если говорит в общем, то прогноз каждого следующего дерева берется с определенным коэффициентом , который обычно равен числу в интервале от 0 до 1. Это нужно для предотвращения переобучения и поиска более сложных закономерностей в данных.

  
Рисунок 10 - Пример работы градиентного бустинга.

Шкала - ошибка на функции потерь.

Проблемой большинства реализаций алгоритма градиентного бустинга является смещение оценки градиента, которая приводит к переобучению. Catboost классификатор исправляет эту проблему при помощи рандомизации на различных этапах создания деревьев. Например, в CatBoost классификаторе используется бутстрапированая подвыборка обучающей выборки для каждого дерева в композиции, что приводит к их рандомизации и, как следствие, избеганию переобучения.

#### **2.3.2 Метод опорных векторов (SVM)**

SVM (support vector machine) - линейный классификатор, максимизирующий расстояние между разделяющей прямой (гиперплоскостью) и крайними точками данных (опорными векторами) по разные стороны от этой прямой (в случае линейной разделимости данных).

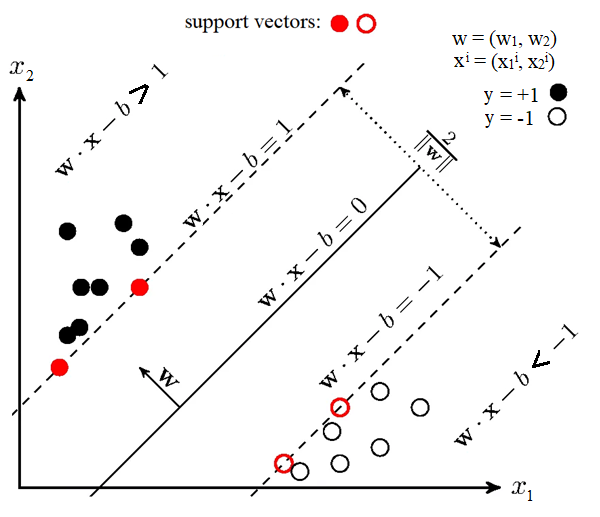


Рисунок 11 - Принцип работы SVM в случае линейно разделимых классов.

Пусть:

- объект, отнесенный к положительному классу с самой низкой уверенностью из возможных (опорный вектор)

- объект, отнесенный к отрицательному классу с самой низкой уверенностью из возможных (опорный вектор)

w - вектор весов классификации SVM

Для разделяющей полосы в случае линейно разделимых выборок справедливо выражение:

, (13)

где вектора ,

оператор <v1, v2> - обозначает скалярное произведение между векторами v1 и v2

Тогда оптимизационная задача имеет следующий вид:

(14)

где обозначает задачу минимизации скалярного произведения вектора w на самого себя.

В случае линейной неразделимой выборки второе условие системы уравнений [15] не может быть выполнено для всех по определению. Для этого задача обобщается на случай линейно неразделимых данных при помощи введения ошибок алгоритма и их интеграции в минимизируемую функцию потерь:

(15)

где C - гиперпараметр, задающий размер штрафов за ошибки.

В нелинейных случаях часто используются ядра. Ядро отображает пространство признаков в новое, где классы становятся почти или полностью линейно разделимы.

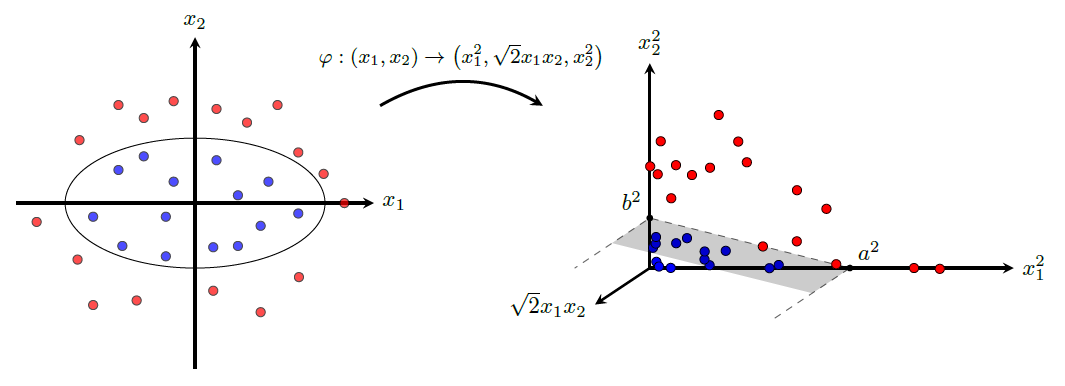


Рисунок 12 - Пример работы спрямляющего пространства в задаче классификации. Предикторы переводятся в новые признаки .

Если вместо скалярного произведения использовать нелинейную симметричную функцию

, (16)

где - спрямляющее отображение.

То метод SVM может быть сформулирован в исходном пространстве с получением нелинейной разделяющей поверхности. В этом суть метода kernel trick.

В качестве ядра может быть использовано полиномиальное ядро, которое представляет собой пространство мночленов не выше степени d:

(17)

#### **2.3.3 Алгоритм Random Forest (случайный лес)**

Случайный лес - ансамблевый метод над простыми деревьями решений (Decision Tree), данный алгоритм машинного обучения исправляет главную проблему дерева решений - сильное переобучение путем усреднения предсказания по множеству деревьев.

Алгоритм основан на композиции решающих деревьев:

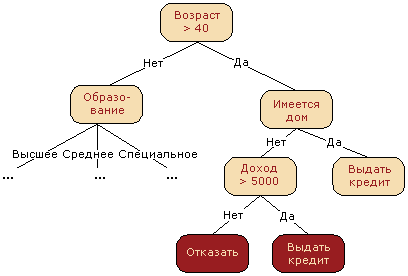


Рисунок 15 - Пример работы решающего дерева

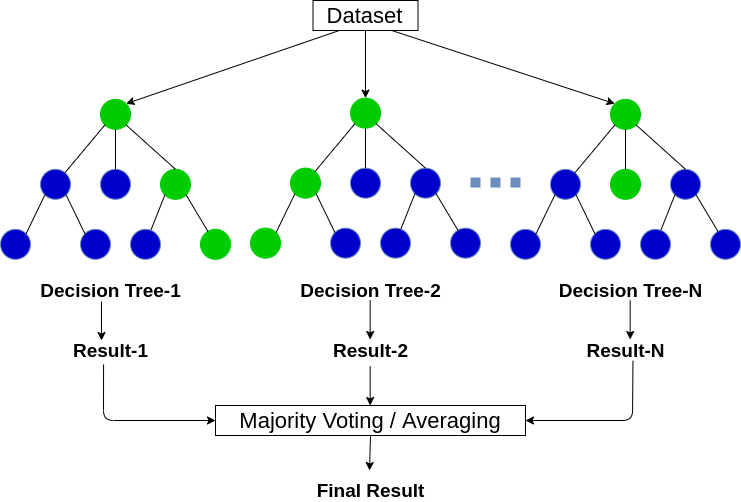


Рисунок 13 - Принцип работы случайного леса

Идея случайного леса состоит в том, чтобы объединить N алгоритмов деревьев решений (от до ) в один, после чего усреднить ответы:

(18)

Однако, для обучения базовых алгоритмов нельзя использовать одну и ту же выборку, причина этого в разложении ошибки в точке x:

, (19)

где:

Квадрат смещения - средняя ошибка алгоритма на всевозможных наборах данных

Дисперсия - дисперсия алгоритма. Различие ответов при обучении на разных наборах данных

- ошибки, которые невозможно устранить (например, погрешности замеров данных)

– обученные алгоритм случайного леса

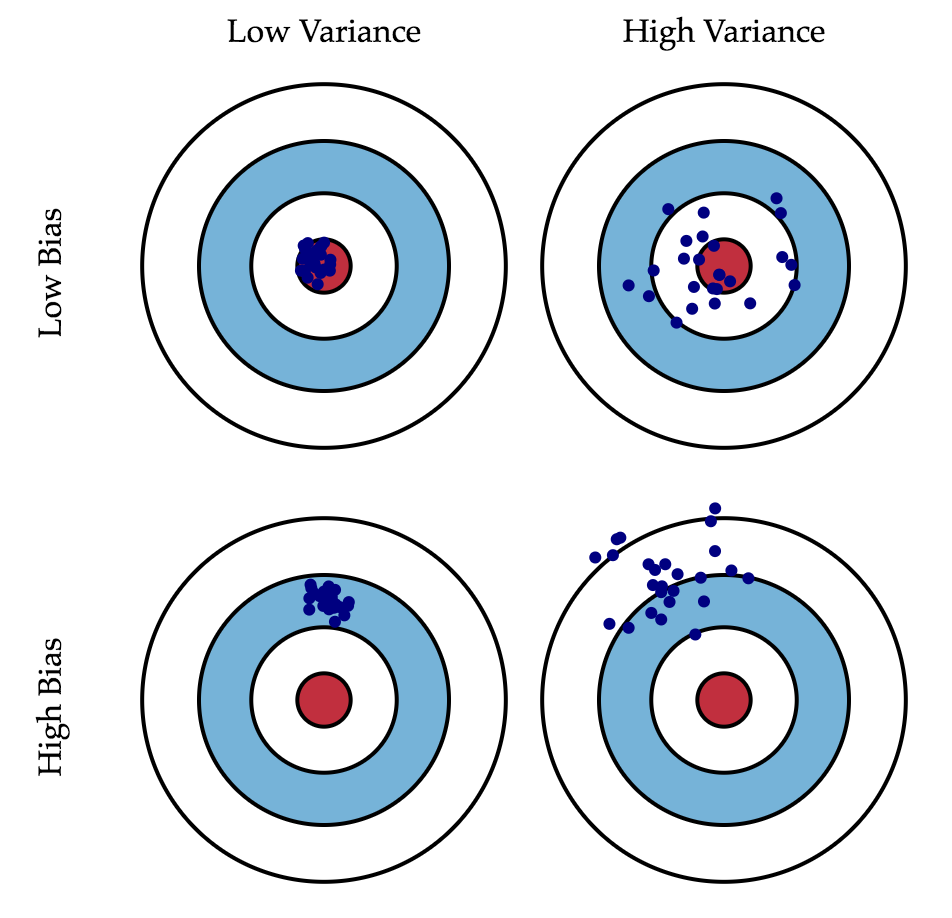


Рисунок 14 - Отличие компонентов ошибки.

Известно, что дерево решений имеет низкий квадрат смещения, но очень высокую дисперсию ошибки, в результате чего он постоянно переобучается. Для того, чтобы исправить эту проблему в случайном лесе используются:

* Бэггинг (бутстрапированная выборка) - каждый базовый алгоритм обучается на своей подвыборке данных, причем, чем более различны эти подвыборки, тем меньше будет дисперсия алгоритма.
* Каждое разбиение в деревьях выбирается из подвыборки всех признаков.

Алгоритм построения случайного леса из N решающих деревьев:

1. Построить с помощью бутстрапа сгенерировать N подвыборки , где n принимает значения от 1 до N
2. Каждая получившаяся подвыборка используется как обучающая для построения соответствующего решающего дерева .
   1. Дерево строится, пока в каждом листе окажется не более объектов. Очень часто деревья строят до конца ( = 1), чтобы получить сложные и переобученные решающие деревья с предельно низким смещением.
   2. Процесс построения дерева рандомизирован на этапе выбора оптимального признака, по которому будет происходить разбиение. Оно ищется среди случайного подмножества размера q, а не среди всего подмножества признаков.
   3. Следует обратить особое внимание, что случайное подмножество размера q выбирается заново каждый раз, когда необходимо разбить очередную вершину. В этом состоит основное отличие такого подхода от метода случайных подпространств, где случайное подмножество признаков выбиралось один раз перед построением базового алгоритма.
3. Построенные деревья объединяются в композицию и ответ алгоритма будет вариант, за который “проголосовало” большинство базовых алгоритмов.

Для построения отдельных деревьев критерий ошибки при каждом разбиении записывается следующим образом:

, (20)

где - критерий информативности,

- количество объектов, ушедших в левый лист,

- количество объектов, ушедших в правый лист.

- общее количество объектов текущего разбиения,

j - номер признака, по которому разбиваются данные,

t - значение признака, по которому бьется выборка .

Рассмотрим критерий, на основании которого совершается разбиение: критерий информативности Джини (Gini):

Если - доля объектов класса в выборке , то рассчитывается он следующим образом:

(22)

Тогда критерий информативности Gini записывается следующим образом:

(21)

Оптимум достигается в случае, когда объекты все относятся к одному классу.

Кроме того, случайный лес способен рассчитывать важность признаков для обучения. Рассмотрим принцип работы:

Чем выше признак разбиения в дереве, тем важнее он для прогноза, поскольку он сильно уменьшает неопределенность в данных (уменьшает критерий Джини).

Для разбиения в дереве решений:

, (22)

где - количество классифицируемых объектов в листе, который является родительским для .

– классифицируемые объекты в листе

- важность отдельного разбиения в дереве по признаку .

Для отдельного дерева в композиции важность признаков рассчитывается как:

(23)

После чего величина нормируется:

(24)

Значимость признаков для всего случайного леса считается по следующей формуле:

, (25)

где T - количество деревьев в случайном лесе.

## **3 Разработка программного обеспечения**

Для реализации программного решения был использован: дистрибутив языка Python - Anaconda 3, так как он является одним из самых мощных инструментов анализа, обработки данных и построения моделей машинного обучения, так как обладает большим количеством высокоэффективных библиотек, предназначенными для этих целей:

Scipy - используется для генерирования новых признаков

Numpy - используется для матричных операций

CatBoost - использование моделей машинного обучения, которые строятся на основе алгоритмов градиентного бустинга

Scikit-learn - используется для построения моделей Random Forest, нормализации данных, деления данных на train, valid, test подвыборки, использования Label Encoder на зависимом столбце и выбора метрик

Pandas - работа с таблицами телеметрии: их форматирование, объединение, удаление столбов, применение метода агрегации данных по извлекаемым статистикам.

Statsmodels - для проверки статистических свойств данных

Seaborn и Matplotlib - библиотеки используется для визуализации confusion matrix, построения диаграммы предсказанных меток и анализа данных.

Mglearn – визуализация матрицы ошибок

*## Подключение библиотек*

import numpy as np

import scipy as sp

import pandas as pd

import matplotlib.pyplot as plt

import matplotlib as mpl

import mglearn

from sklearn.preprocessing import MinMaxScaler, RobustScaler, LabelEncoder, StandardScaler

from scipy.stats import normaltest, probplot, poisson, chisquare, shapiro, ttest\_ind, chi2\_contingency

import scipy

from scipy import stats

import statsmodels.api as sm

from sklearn.model\_selection import cross\_val\_score

plt.rc("font", family="Verdana") # кириллица

from sklearn.feature\_selection import f\_classif

from sklearn.ensemble import RandomForestClassifier

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

from sklearn.metrics import classification\_report, confusion\_matrix, roc\_auc\_score

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

import catboost as cb

from sklearn.metrics import log\_loss

*## Считывание изучаемых данных*

table\_Pasha = pd.read\_excel("./datasets/Разметка паша.xlsx")

table\_Max = pd.read\_excel("./datasets/Разметка макс.xlsx")

table\_Pasha.columns = table\_Max.columns = ["Дата", "Дата\_получения\_локации", "Широта", "Долгота", "Высота", "Скорость", "Точность", "Ax",Ay", "Az", "Сердечный\_Ритм", "Шаги", "Разметка"]

print("Измененные колонки для \nПервого датасета {}, \nВторого датасета {}".format(list(table\_Pasha.columns),

list(table\_Max.columns)))

*## Статистические характеристики изучаемых данных*

**with** pd.option\_context('display.max\_rows', **None**, 'display.max\_columns', **None**):

print("Данные первого датасета:\n{}\n\n\nДанные второго датасета\n{}"

.format(table\_Max.describe(), table\_Pasha.describe()))

*## Распределение признака “Дата\_получения\_локации” в датасетах*

plt.figure(figsize=(15, 7))

table\_Pasha.Дата\_получения\_локации.plot(label="Второй датасет")

table\_Max.Дата\_получения\_локации.plot(label="Первый датасет")

plt.legend();

plt.xlabel("Количество точек данных")

plt.ylabel("Дата со спутников");

*## Распределение признака “Дата” в датасетах*

plt.figure(figsize=(15, 7))

table\_Pasha.Дата.plot(label="Второй датасет")

table\_Max.Дата.plot(label="Первый датасет")

plt.legend();

plt.xlabel("Количество точек данных")

plt.ylabel("Дата");

*## Q-Q графики для признаков: “Ax”, “Ay”, “Az”, “Сердечный\_ритм”*

corr\_table = pd.concat([table\_Pasha.drop(["Разметка", "Шаги"], axis=1), table\_Max.drop(["Разметка", "Шаги"], axis=1)], axis=0)

fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(16, 10))

**for** idx, x **in** enumerate(corr\_table):

pp = sm.ProbPlot(corr\_table[x].values, fit=**True**);

pp.qqplot(line="45", ax=axes[idx // 2, idx % 2])

axes[idx // 2, idx % 2].set\_xlabel("Наблюдаемые квантили признака {}".format(x))

axes[idx // 2, idx % 2].set\_ylabel("Ожидаемые квантили признака {}".format(x))

*## Признак “Шаги” в двух датасетах*

plt.figure(figsize=(15, 7))

table\_Pasha.Шаги.plot(label="Второй датасет")

table\_Max.Шаги.plot(label="Первый датасет")

plt.legend();

plt.xlabel("Количество точек данных")

plt.ylabel("Количество шагов");

*## Перекодирование меток классов в соответствии с таблицей 9*

**def** **change\_class\_inplace**(df):

df.replace({'Разметка': 'c'}, 'e', inplace=**True**)

df.replace({'Разметка': 'd'}, 'c', inplace=**True**)

df.replace({'Разметка': 'f'}, 'e', inplace=**True**)

df.replace({'Разметка': 'g'}, 'd', inplace=**True**)

df.replace({'Разметка': 'h'}, 'f', inplace=**True**)

df.replace({'Разметка': 'i'}, 'd', inplace=**True**)

df.replace({'Разметка': 'j'}, 'c', inplace=**True**)

df.replace({'Разметка': 'k'}, 'a', inplace=**True**)

loss = df.query('Разметка == "l"')

df.drop(df[df.Разметка == "l"].index, inplace=**True**)

df.reset\_index(drop=**True**, inplace=**True**)

**return** loss

pasha\_loss = change\_class\_inplace(table\_Pasha)

max\_loss = change\_class\_inplace(table\_Max)

*## Удаление неинформативных признаков*

table\_Pasha.drop(axis=1, inplace=**True**, columns=["Широта", "Долгота", "Дата", "Высота", "Скорость",

"Точность", 'Дата\_получения\_локации'])

table\_Max.drop(axis=1, inplace=**True**, columns=["Широта", "Долгота", "Дата", "Высота", "Скорость",

"Точность", 'Дата\_получения\_локации'])

*## Обработка признака “Шаги”*

table\_Pasha\_new = table\_Pasha.copy()

table\_Max\_new = table\_Max.copy()

p\_f\_u = [int(x) **for** x **in** table\_Pasha.Шаги.unique()]

m\_f\_u = [int(x) **for** x **in** table\_Max.Шаги.unique()]

**def** **foot**(c, f\_u):

c = int(c)

**if** c != f\_u[0]:

**return** c - f\_u[f\_u.index(c) - 1]

**else**:

**return** f\_u[0]

table\_Pasha\_new["Шаги"] = table\_Pasha.Шаги.apply(foot, args=(p\_f\_u,))

table\_Max\_new["Шаги"] = table\_Max.Шаги.apply(foot, args=(m\_f\_u,))

table\_Pasha\_new.replace({"Шаги": 502}, 2, inplace=**True**)

*## Нормализация датасетов*

scaled\_features = StandardScaler().fit\_transform(table\_Pasha\_new.drop(["Разметка"], axis=1))

table\_Pasha\_new\_s = pd.concat([pd.DataFrame(scaled\_features,

columns=table\_Pasha\_new.drop(["Разметка"], axis=1).columns),

table\_Pasha\_new[["Разметка"]]], axis = 1)

scaled\_features = StandardScaler().fit\_transform(table\_Max\_new.drop(["Разметка"], axis=1))

table\_Max\_new\_s = pd.concat([pd.DataFrame(scaled\_features,

columns=table\_Max\_new.drop(["Разметка"], axis=1).columns),

table\_Max\_new[["Разметка"]]], axis = 1)

*## Генерация новых признаков*

**def** **agg\_interval**(df, sample\_time="5S"):

**def** **mode\_int**(x):

**return** stats.mode(x).mode

**def** **quantile\_25**(x):

**return** np.quantile(x, .25)

**def** **quantile\_75**(x):

**return** np.quantile(x, .75)

**def** **str\_func**(x):

**if** len(x) <= 1:

**return** "0" + x

**else**:

**return** x

V = ""

**def** **mode\_str**(x):

**nonlocal** V

mode = str(\*stats.mode(list(x)).mode)

**if** **not** mode:

mode = V

**else**:

V = mode

**return** mode

indexx = df.reset\_index().index

hours = indexx // 3600

minutes = indexx // 60 - (indexx // 3600) \* 60

seconds = indexx - hours \* 3600 - minutes \* 60

hms = pd.DataFrame({'hour': hours,

'minute': minutes,

'second': seconds})

df = pd.concat([pd.DataFrame(pd.to\_datetime(hms.iloc[:, 0].astype(str).apply(str\_func) \

+ ":" + hms.iloc[:, 1].astype(str).apply(str\_func) \

+ ":" + hms.iloc[:, 2].astype(str)\

.apply(str\_func), format="%H:%M:%S"), columns=["Date"]), df], axis=1)

d = {'Разметка': mode\_str}

**for** x **in** set(df.columns) - set(["Разметка", "Date"]):

d[x] = [mode\_int, quantile\_25, quantile\_75, np.mean, np.std, stats.skew, stats.kurtosis, \

np.min, np.max, stats.entropy]

new\_df = df.set\_index("Date").resample(sample\_time).agg(d)

print("Количество пропущенных значений (столбец):", new\_df.iloc[:, 3].isnull().values.sum())

print("Всего значений (столбец):", new\_df.shape[0])

**for** i **in** new\_df.columns: *# итерируемся по столбцам*

new\_df[i].fillna(method="ffill", inplace=**True**)

**return** new\_df

table\_Pasha\_new\_s\_feature = agg\_interval(table\_Pasha\_new\_s)

table\_Max\_new\_s\_feature = agg\_interval(table\_Max\_new\_s)

*## Соединение двух обработанных таблиц*

table\_s\_feature = pd.concat([table\_Pasha\_new\_s\_feature, table\_Max\_new\_s\_feature], axis=0)

*## Разделение данных на train, valid, test подвыборки*

table\_s\_feature = table\_s\_feature.reset\_index(drop=**True**)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(clean\_dataset(table\_s\_feature.drop("Разметка", axis=1)),

table\_s\_feature["Разметка"],

random\_state=1, test\_size=.15)

X\_train, X\_valid, y\_train, y\_valid = train\_test\_split(X\_train, y\_train, random\_state=2, test\_size=.18)

*## Обучение RF*

y\_enc = ['a', 'b', 'd', 'c', 'e']

enc = LabelEncoder()

enc.fit(y\_enc)

y\_train\_rf = pd.DataFrame(enc.transform(y\_train.values.ravel()), columns=["Разметка"])

y\_valid\_rf = pd.DataFrame(enc.transform(y\_valid.values.ravel()), columns=["Разметка"])

forest = RandomForestClassifier(n\_jobs=-1, random\_state=1, n\_estimators=500).fit(X\_train, y\_train\_rf.values.ravel())

print("Правильность на валидационном наборе: {:.2f}".format(forest.score(X\_valid, y\_valid\_rf.values.ravel())))

print(classification\_report(forest.predict(X\_valid), y\_valid\_rf))

score\_image = mglearn.tools.heatmap(

confusion\_matrix(y\_valid\_rf, forest.predict(X\_valid)),

xlabel="Спрогнозировання метка класса",

ylabel="Фактическая метка класса",

xticklabels=sorted(enc.classes\_[sorted(y\_valid\_rf["Разметка"].unique())]),

yticklabels=sorted(enc.classes\_[sorted(y\_valid\_rf["Разметка"].unique())]),

cmap=plt.cm.gray\_r, fmt="%d")

*## 10 важнейших признаков для обучения RF*

d = {}

**for** i **in** zip(X\_valid.columns, forest.feature\_importances\_):

d[i[0]] = i[1]

d = dict(sorted(d.items(), key=**lambda** item: item[1], reverse=**True**))

i1 = 0

**for** i **in** d:

i1 += 1

print("Важность признака: {} для распознавания составляет: {}".format(i, d[i]))

**if** i1 == 10:

**break**

*## Обучение CatBoost классификатора*

model = cb.CatBoostClassifier(task\_type="GPU")

model.fit(X\_train,y\_train, eval\_set=(X\_valid, y\_valid), plot=**True**, verbose=**False**)

*## Тестирование CatBoost на валидационной выборке*

print("Правильность на валидационном наборе: {:.2f}".format(model.score(X\_valid, y\_valid)))

print(classification\_report(model.predict(X\_valid), y\_valid))

plt.figure()

score\_image = mglearn.tools.heatmap(

confusion\_matrix(y\_valid, model.predict(X\_valid)),

xlabel="Спрогнозировання метка класса",

ylabel="Фактическая метка класса",

xticklabels=sorted(y\_valid.mode\_str.unique()),

yticklabels=sorted(y\_valid.mode\_str.unique()),

cmap=plt.cm.gray\_r, fmt="%d")

*## 10 важнейших признаков для обучения CatBoost классификатора*

test\_pool = cb.Pool(X\_valid, y\_valid, feature\_names=list(X\_valid.columns))

d = {}

**for** i **in** zip(X\_valid.columns, model.feature\_importances\_):

d[i[0]] = i[1]

d = dict(sorted(d.items(), key=**lambda** item: item[1], reverse=**True**))

i1 = 0

**for** i **in** d:

i1 += 1

print("Важность признака: {} для распознавания составляет: {}".format(i, d[i]))

**if** i1 == 10:

**break**

*## Тестирование RF на тестовой выборке*

y\_enc = ['a', 'b', 'd', 'c', 'e']

enc = LabelEncoder()

enc.fit(y\_enc)

y\_test\_rf = pd.DataFrame(enc.transform(y\_test.values.ravel()), columns=["Разметка"])

print("Правильность на тестовом наборе: {:.2f}".format(forest.score(X\_test, y\_test\_rf.values.ravel())))

print(classification\_report(forest.predict(X\_test), y\_test\_rf.values.ravel()))

score\_image = mglearn.tools.heatmap(

confusion\_matrix(y\_test\_rf.values.ravel(), forest.predict(X\_test)),

xlabel="Спрогнозировання метка класса",

ylabel="Фактическая метка класса",

xticklabels=sorted(enc.classes\_[sorted(y\_test\_rf["Разметка"].unique())]),

yticklabels=sorted(enc.classes\_[sorted(y\_test\_rf["Разметка"].unique())]),

cmap=plt.cm.gray\_r, fmt="%d")

*## Результирующая статическая характеристика классификации меток*

f\_p = enc.inverse\_transform(forest.predict(X\_test))

print("Декодирование меток классов:\na - простой, b - перемещение, \nc - низкая мобильная активность" +

"\nd - высокая мобильная активность\ne - низкая статическая активность,\nf - высокая статическая активность")

print("Метка - встречаемость:\n{}\n".format(pd.value\_counts(f\_p)))

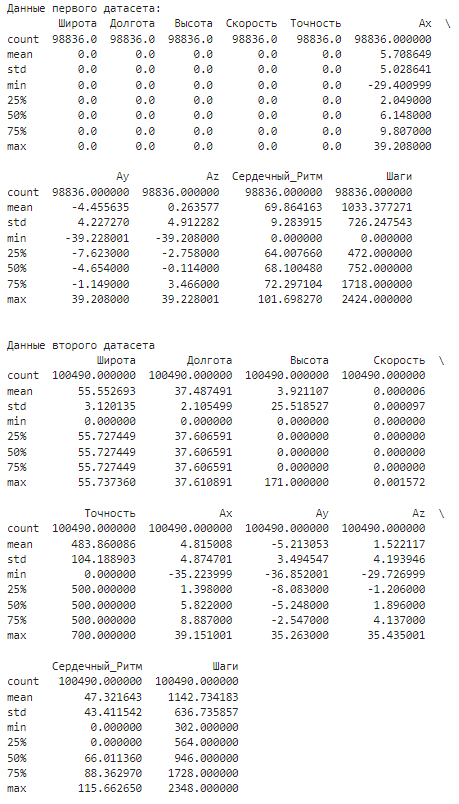
sns.histplot(f\_p);

plt.xlabel("Метки рабочей активности сотрудников")

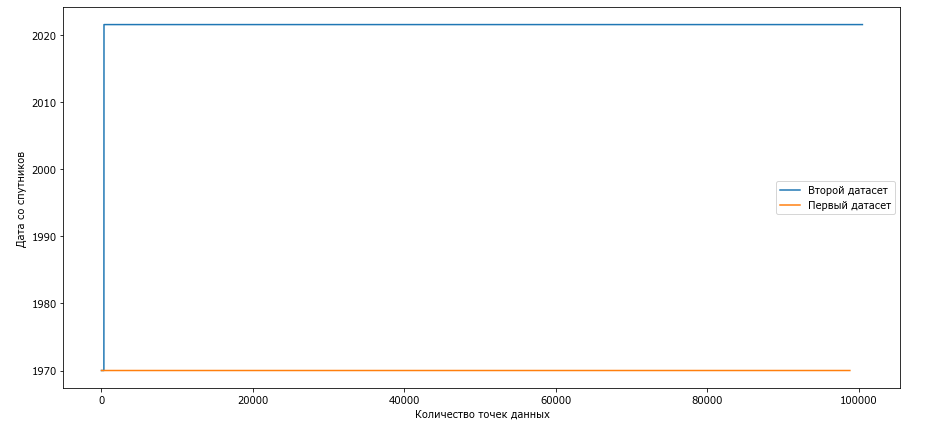
plt.ylabel("Количество меток");

## **4 Компьютерное моделирование**

Загрузим первоначальные данные. Выведем статистические признаки:

  
Рисунок 15 - Статистические характеристики изучаемых данных.

Как видно из сводной таблицы, признаки “Широта”, “Долгота”, “Высота”, “Точность” “Скорость” не имеют в первом датасете значений, отличных от нуля. Кроме того, их связь с активностью рабочих ставится под сомнение в виду смыслового значения этих признаков, за исключением признака “Скорость”. К сожалению, cохранение признака “Скорость” также не имеет смысла, поскольку в обоих датасетах его значения практически все нулевые. Признаки, кодирующие дату, как правило, используются для агрегации данных. Выведем графики признаков “Дата” и “Дата\_получения\_локации”:

  
Рисунок 16 – Распределение признака “Дата\_получения\_локации” в датасетах.

Как видно из графика, этот признак можно не использовать для дальнейшей обработки, так как он не содержит полезной информации. Рассмотрим признак “Дата”:

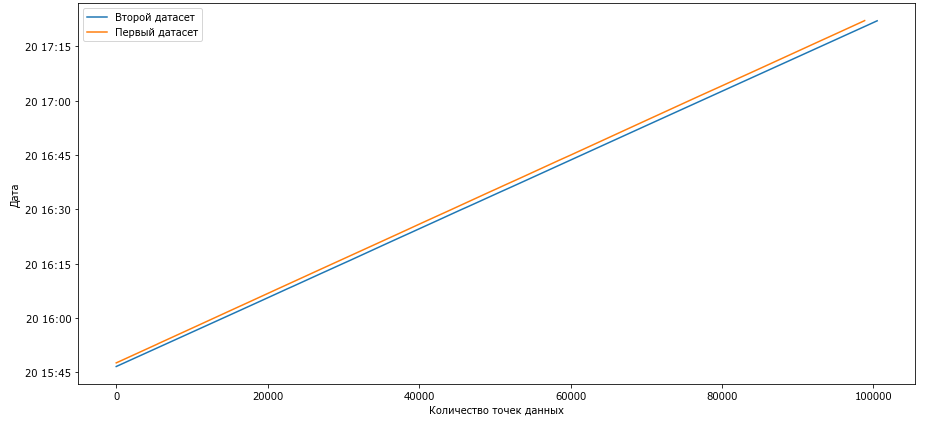
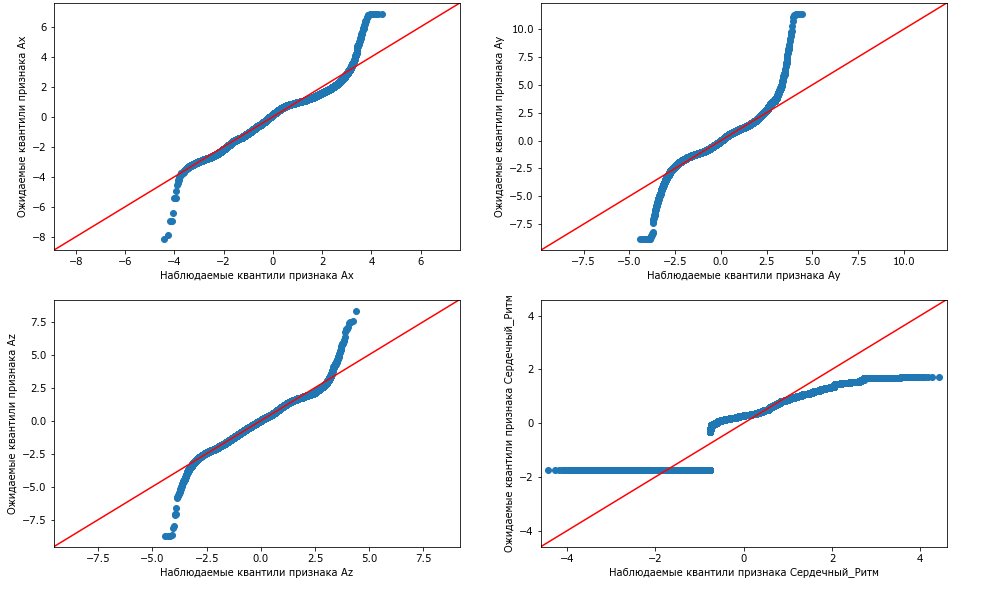


Рисунок 17 – Распределение признака “Дата” в датасетах.

Как видно из графика, данные двух датасетов записывались равномерно на протяжении некоторого времени. Временной масштаб не соответствует действительному, а значит считан устройством мониторинга неправильно. Однако, для агрегации данных можно использовать информацию о равномерности записи данных. Этой информации для дальнейшей работы достаточно и сохранять признак не имеет смысла.

Кроме того, в данных второго датасета 25 квантиль признака “Сердечный\_ритм” равен нулю. Для большей наглядности рассмотрим признаки “Ax”, “Ay”, “Az”, “Сердечный\_ритм” на графике в объединении по датасетам на Q-Q графике:

  
Рисунок 18 - Q-Q графики для признаков: “Ax”, “Ay”, “Az”, “Сердечный\_ритм”.

Наблюдаемые данные сравнивались с ожидаемыми. Ожидаемые данные строились относительно предположения о нормально данных изначальных. Видно, что наблюдаемые данные слабо походят на нормально распределенные. Признак “Сердечный\_ритм” имеет чуть меньше половины значений, равных нулю, что крайне плохо для предсказания активности рабочих.

Рассмотри признак “Шаги” отдельно:

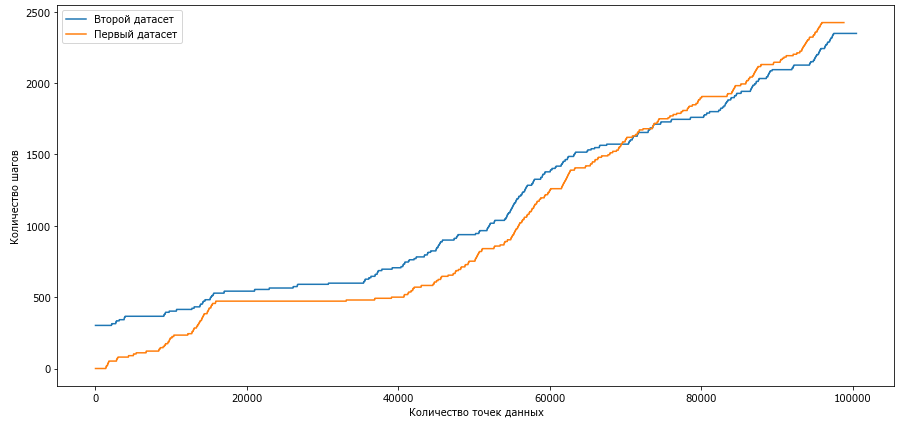


Рисунок 19 - Признак “Шаги” в двух датасетах.

Проведем форматирование признаков:

Перекодируем метки классов, в соответствии с таблицей 9. Удалим из данных те признаки, которые признаны неинформативными в соответствии с анализом данных. Обработаем признак “Шаги”. После чего нормализуем датасеты.

Для извлечение использована агрегация данных в окне, которое равно 5 строкам. Извлекаемые признаки: 25 и 75 квантили, математическое ожидание, среднеквадратичное отклонение, коэффициент асимметрии, эксцесса минимум, максимум и энтропия. Для обработки зависимого признака, в случае попадания нескольких меток классов в один интервал, используется мода, которая выбирает ту метку, которая чаще всего встречалась. Признак, по которому осуществляется агрегация, был сгенерирован из индексов строк данных. Размер окна составил 5 строк.

Обучим модель Random Forest на train подвыборке и проверим ее качество на валидационной:

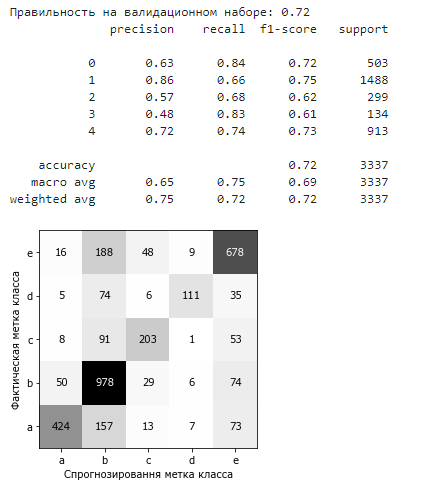


Рисунок 20 - Результаты обучения RF на валидационной выборке.

Результаты получились крайне высокими, посмотрим 10 самых важных признаков для предсказания:

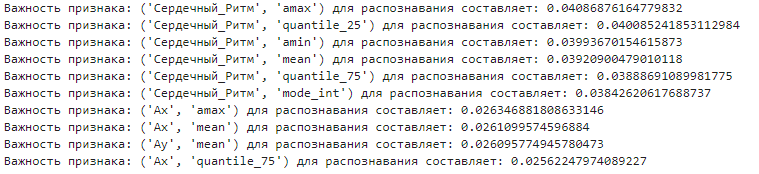


Рисунок 21 - 10 важнейших признаков для обучения RF.

Обучим CatBoost классификатор:

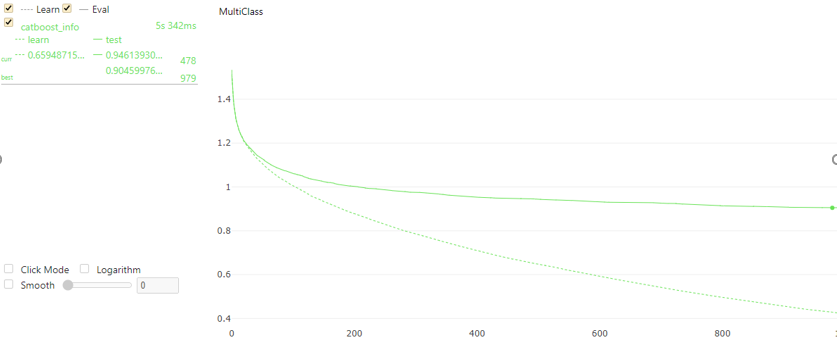


Рисунок 22 - Процесс обучения CatBoost классификатора. В качестве функции ошибки показана LogLoss.

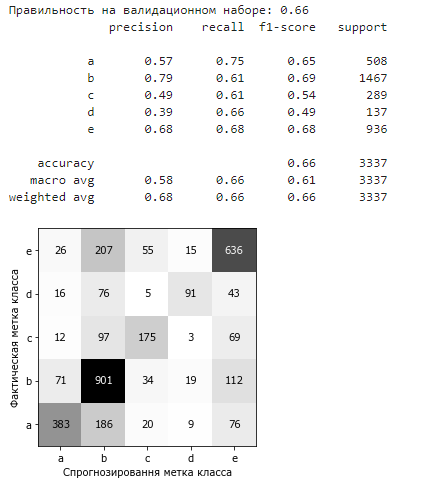


Рисунок 23 - Результаты обучения CatBoost классификатора на валидационной выборке.

10 важнейших признаков для обучения CatBoost представлены на рисунке 31.

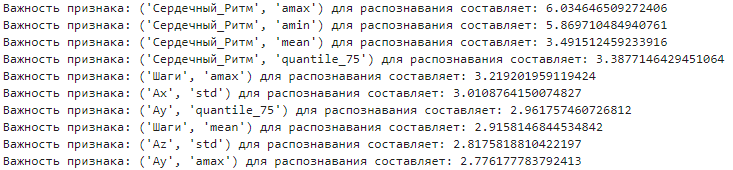
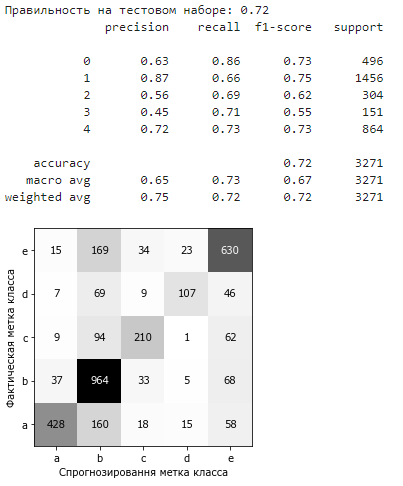


Рисунок 24 - 10 важнейших признаков для обучения CatBoost классификатора.

Как видно из результатов обучения, Random Forest в данной задаче оказался эффективней CatBoost классификатора.

Качество на тестовой (отложенной) выборке:

  
Рисунок 25 - Результаты случайного леса на тестовой выборке.

Результирующая статистика может отправляться в следующем виде:

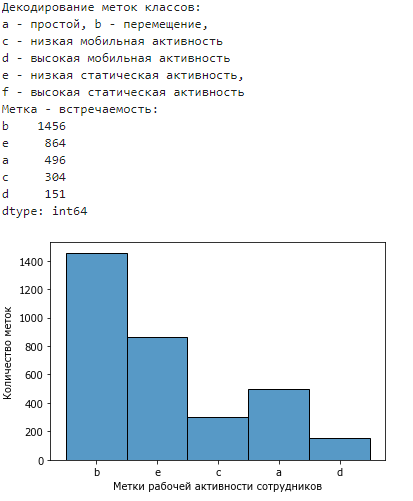


Рисунок 26 - Результирующая статическая характеристика классификации меток.

## **5 Анализ результатов компьютерного моделирования**

Были проведены анализ и обработка собранной телеметрии рабочих, сравнивалось качество моделей машинного обучения, по результатам которого выбиралась наилучшая модель. После чего ее качество измерялось на тестовой подвыборке и строилась гистограмма распределения меток, которую можно использовать управляющими низшего звена для анализа производственной активности.

Данные, считанные с устройств мониторинга, записывались в условиях реальной работы и размечались вручную. По этим причинам датасеты содержат много аномалий, особенно в признаках, кодирующих временную информацию и сердцебиение человека. Однако, алгоритм случайного леса дал точность прогноза, равную 72% на отложенной выборке. При уменьшении количества аномалий в данных и большем количестве телеметрии точность предсказания повысится значительно.

Способы анализа и обработки данных, предложенные в данной работе, а также результаты сравнения моделей CatBoost и Random Forest в задаче классификации целенаправленной физической активности могут быть полезны для будущих работ по схожим темам.