Документация

Описание алгоритма и программного кода для создания и использования прогнозных моделей отказов тяговых электродвигателей электровозов 2ЭС6.

Разработанная программная библиотека состоит из трёх модулей: модуля подготовки данных в табличном формате для обучения моделей, модуля, предназначенного для обучения и оценки качества построенных моделей и модуля, предназначенного для прогнозирования отказов тяговых двигателей с использованием обученных моделей. Данные модули запускаются при помощи программного интерпретатора python версии 3.6.0 или выше, скачать который можно по адресу https://www.python.org/downloads/. Для запуска данных модулей может быть необходима установка некоторых библиотек, перечисленных в заголовке файлов, содержащих программный код соответствующих модулей.

Программный код модуля подготовки данных в табличном формате для обучения моделей содержится в файле prepare.py. Данный модуль использует предобработанные данные, полученные при помощи программного модуля parse.py, разработанного в рамках предыдущего этапа. Путь к предобработанным данным задаётся переменной input_path. При запуске программного модуля создаётся (только если переменная set_header равна True – в случае повторного запуска алгоритма для обновления существующей таблицы необходимо присвоить этой переменной значение False, иначе данные запишутся поверх существующих) файл (его имя задаётся переменной output_file, а находится он в папке, заданной переменной output_path), в который буду записаны строки таблицы, и в него записываются имена столбцов. После этого вызывается функция parse_directory, обрабатывающая все

предобработанные файлы в директории со входными данными. Имена уже обработанных файлов хранятся в файле, определяемой переменной done_file и не обрабатываются при повторном запуске алгоритма для обновления существующей таблицы. Функция обработки отдельного предобработанного файла называется parse_file. Вначале она определяет границы фрагментов данных, содержащихся в файле. Далее, для каждой фрагмента и каждой секунды внутри него определяются значения всех аналоговых контролируемых параметров (определённых в переменной params), затем вычисляется сумма этих значений внутри фрагмента (для некоторых параметров, определённых в массиве absolute_parameters, значения суммируются по абсолютному значению), а также продолжительность самого фрагмента. Если разделить первое значение на второе получится среднее значение параметра внутри фрагмента. Также вычисляется количество отказов внутри фрагмента (определённых как снижение параметра, определённого в массиве target_parameters, ниже уставки, определённой переменной target_constraint в течение времени в секундах, определённого переменной target_time) и суммарное время, когда значение таких параметров находится ниже соответствующих уставок.

Далее, фрагменты данных последовательно (по времени) считываются, и из них складываются новые фрагменты либо заданной продолжительности (определяемой переменной time_unit – по умолчанию, не менее 24 часов непрерывной работы), либо отстоящих от последующего фрагмента на заданный временной промежуток (определяемой переменной time_horizon). Для каждого такого фрагмента определяется наличие или отсутствие внутри него отказа, и с использованием фрагментов, полученных ранее и находящихся во временных промежутках, определённых переменной borders, вычисляются средние значения аналоговых контролируемых параметров (а также среднее количество отказов на единицу времени(признак с

префиксом ‘fails_’) и отношение времени отказа к общему времени фрагмента (признак с префиксом ‘fails_time_’)) внутри каждого временного промежутка и заносятся в таблицу вместе с маркером наличия или отсутствия отказа в текущем фрагменте (признак с префиксом ‘fail_’), номером и секцией локомотива(признак ‘train’), временного промежутка, внутри которого усредняются контролируемые параметры, его продолжительности и времени окончания (признаки time_borders, features_duration и features_end_time соответственно), продолжительности текущего временного промежутка (признак target_duration).

Программный код модуля, предназначенного для обучения и оценки качества построенных моделей содержится в файле train.py. В самом начале он загружает из файла, определённого переменной input_file, таблицу данных, простроенную при помощи предыдущего модуля, и сортирует её по времени окончания фрагмента данных, используемого для прогноза. В зависимости от установленной булевой переменной select_best_columns, признаки, являющиеся лучшими для разных моделей, либо загружаются из текстового файла, определённого переменной bestcols_file (если select_best_columns == False), либо определяются во время выполнения кода программного модуля с помощью метода RFECV из библиотеки sklearn и записываются в тот же файл (если select_best_columns == True). Признаки, определённые в массиве main_columns используются во всех моделях вне зависимости от того, были ли они отобраны. Для каждого горизонта прогнозирования отказов и пары ТЭД строится отдельная модель. Данные разделяются на обучающую и тестовую выборки в пропорциях, установленной переменной train_test_split. Модели обучаются и тестируются при помощи методов логистической регрессии из библиотеки sklearn и градиентного бустинга над решающими деревьями из библиотеки xgboost. Для каждого метода рассчитывается и записывается в файл (имя

которого определено переменной results_file) метрика roc-auc для обучающей и тестовой выборок, для логистической регресси дополнительно рассчитываются и записываются метрики accuracy и f-мера. Кроме того, при помощи обученной модели логистической регрессии в файл результата записываются признаки по убыванию значимости. После этого модели, обученные на полном массиве данных, сохраняются в соответствующие файлы. Все файлы сохраняются в директорию, определённую переменной output_path.

Программный код модуля, предназначенного для прогнозирования отказов тяговых двигателей с использованием обученных моделей содержится в файле test-server.py. Данный программный модуль представляет собой сервер, после получения соответствующего GET-запроса возвращающий JSON объект, содержащий результаты прогноза. Пример строки GET- запроса: ‘http://127.0.0.1:8090/test?lokomotiv=543_A&date=2017-02-05&distance=7&target=1-2’, где в переменной lokomotiv передается номер и секция соответствующего локомотива, в переменной date – дата, от которой строится прогноз, в переменной distance – горизонт прогнозирования, в переменной target – пару ТЭД, для которой строится прогноз. В соответствии с переданными при помощи GET-запроса данными и аналогично алгоритму, использованному в первом модуле, производится выборка и усреднение данных. Далее, загружаются полученные при выполнении второго модуля модели, и при их помощи строится и записывается в JSON объект прогноз. Также в JSON объект записываются признаки (средние значения параметров), по которым был построен прогноз и графики по четырем главным признакам, определённым в переменной visual_parameters. На графиках отображены как точки, взятые из обучающей выборки (красным цветом отмечаются отказы, зелёным – их отсутствие), так и точка (синим цветом), отражающая признаки, использованные для построения прогноза.

1. Описание API

Получение результатов

URL*: host/test?lokomotiv=...&date=...&distance=..&target=...*

Type: GET

Параметры запроса:

lokomotiv: string - название локомотива

date: string - дата от которой считается прогноз

target: string - ТЭД 1-2 или ТЭД 3-4

distance: string - время в течение которого строиться прогноз (1, 7, 14, 30 дней)

Пример: *host/test?lokomotiv=543_A&date=2017-02-05&distance=7&target=1-2*

Ответ сервера: Json-объект вида

{

parameters: {} - *json объект ключевых факторов, влияющих на прогноз*

pictures: [,…] - *массив графиков в формате base64*

prediction: *string - статус прогноза*

probability: *string - вероятность отказа*

}

Пример:

{

parameters: {Ftz: "3.5269406337497187", Fтел2 [кН]: "3.933959598493216", Iвозб. 3-4: "33.187742045641706",…}

pictures: [,…]

prediction: "Нет отказа"

probability: "0.45124536025701184"

}

2. Развертывание сервера

Аппаратные требования

Для развертывания и запуска web-сервера, минимальные аппаратные требования: 6 CPU Core, 16Gb RAM, объем постоянной памяти для функционирования сервера - необходимый для хранения исторических данных

Программные требования

Для запуска сервера необходимо установить

- Python версии 3.6.0
- Анаконда