МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

09.04.03 Прикладная информатика
(код и наименование направления подготовки)

Информационные системы и технологии корпоративного управления
(направленность (профиль))

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ)

на тему «Исследование методов машинного обучения в оценке трудоемкости разработки корпоративных информационных систем»

Студент М. Ю. Хвостова
(И.О. Фамилия) (личная подпись)

Научный руководитель доцент Е. А. Ерофеева
(ученая степень, звание, И.О. Фамилия)

Тольятти 2021

Оглавление

Введение

За последние десятилетия программные системы становились все более сложными. Информационные технологии оказывают влияние на все области знаний. В связи с усложнением программного обеспечения возрастает важность исследований и создание методов оценки трудоемкости разработки программного продукта. Оценка количества усилий, необходимых для разработки программного обеспечения, является важной задачей управления проектом. Трудоемкость разработки напрямую связана с составлением бюджета и планированием проектов. Точная оценка программного обеспечения влияет на успех проекта. Ошибки, допущенные при оценке, приводят к перерасходу средств на проект, задержке выдачи решений, закрытию проектов. Все это сказывается на репутации компании на рынке. Библиографический анализ, сделанный Янгом и другими [1], обнаруживает тревожные данные: от 59% до 76% проектов выполняются с усилием, превышающим начально прогнозируемые сроки. И от 35% до 80% доставляются с опозданием.

По данным компании International Data Corp, занимающейся аналитикой рынка ИТ, расходы организаций на автоматизацию бизнеса к 2022 году достигнут 1,97 триллиона долларов США [2]. По этой причине цена ошибки на этапе планирования проекта может принести бизнесу большие убытки. В отчете PMI (Project Management Institute, Институт управления проектами) - американской некоммерческой профессиональной организации по управлению проектами, в 2016 году на каждый миллиард долларов, вложенных в проекты в США, 122 миллиона долларов были потрачены впустую из-за некачественного планирования и выполнения проекта [3]. По отчету за 2017 год, 27% проектов превысили запланированный бюджет [4]. В 2018 году 9,9% каждого доллара тратится

впустую из-за плохой работы проекта - это 99 миллионов долларов на каждый вложенный миллиард долларов [5].

В связи с этим, возникает потребность в создании системы для оценки трудоемкости разработки программного обеспечения корпоративных информационных систем с использованием методов машинного обучения, которая предоставляет модель для прогнозирования затрат на исполнение программного продукта. В отчете PMI за 2019 год 85% организаций считают, что искусственный интеллект для управления проектами значительно изменят их методы ведения бизнеса в ближайшие пять лет [6]. Использование методов машинного обучения обусловлено следующими преимуществами: возможность обрабатывать большие наборы данных, возможность вовремя заметить и проанализировать изменения в процессе, возможность объективно и точно оценить результат, возможность свести к минимуму ошибки, возникающих в результате человеческого фактора.

Таким образом, получение качественных результатов прогнозирования трудоемкости разработки является актуальной задачей на сегодняшний день. Это будет полезно как бизнесу, предоставляя инструмент для прогнозирования трудоемкости затрат и оценки планируемого бюджета, так и исполнителям, осуществляя анализ существующих проектов компании и на основе анализа прогнозируя затраты на новые проекты.

Целью работы является исследование методов машинного обучения в оценке трудоемкости разработки программного обеспечения корпоративных информационных систем.

Объектом исследования является процесс оценивания трудоемкости разработки программного обеспечения на основе имеющихся исторических данных о проектах, представленных в виде датасетов. Предметом исследования являются методы машинного обучения для прогнозирования трудоемкости разработки программного обеспечения.

Для достижения поставленной цели необходимо решить следующие задачи:

- провести обзор методов машинного обучения для оценки трудоемкости разработки корпоративных систем,
- провести предварительный эксперимент на данных с использованием выбранных методов машинного обучения,
- определить оптимальный метод для прогнозирования трудоемкости с использованием классических методов машинного обучения, нейронных сетей и ансамблей методов, который позволяет обеспечить точность прогнозирования трудоемкости с наименьшим значением средней квадратичной ошибки (MSE),
- спроектировать веб-систему,
- реализовать веб-систему.

Гипотеза: исследование будет более эффективно, если

1. будут рассмотрены существующие методы машинного обучения для оценки трудоемкости разработки корпоративных систем;
2. будут определены наиболее эффективные методы для оценки трудоемкости, а также найдены их недостатки;
3. будут рассмотрены существующие корпоративные системы для оценки трудоемкости разработки;
4. будут определены преимущества и недостатки;
5. будет создана модель на основе проведенного анализа с учетом проанализированных методов и существующих корпоративных систем, выявленных их достоинств и недостатков;
6. будут выбраны технологии, с помощью которых будет реализована модель;
7. она будет реализована;
8. будет проведен эксперимент на данных с использованием созданных моделей.

Научная новизна состоит в том, что будет реализован новый ансамблевый метод для решения поставленной задачи из наиболее эффективных методов машинного обучения.

Основные положения, выносимые на защиту:

1. Модель на основе ансамблевого метода машинного обучения, показывающая наиболее эффективный результат для решения задачи оценки трудоемкости разработки корпоративных информационных систем.
2. Результат применения созданной модели в веб-системе, показывающий эффективность и корректность ее реализации.

Объем и структура диссертации: диссертация состоит из введения, четырех глав, заключения и библиографии (33 наименования). Работа содержит 68 страниц, 15 рисунков, 11 таблиц.

Основные результаты исследования представлены в следующих публикациях:

1. Хвостова М.Ю. Обзор алгоритмов для решения задачи оценки трудоемкости разработки программного обеспечения корпоративных систем // Студенческий вестник: электронный научный журнал. 2021. № 3(148).
2. Хвостова М. Ю. Применение алгоритмов для решения задачи оценки трудоемкости разработки программного обеспечения корпоративных систем // Молодой исследователь: вызовы и перспективы: сб. ст. по материалам CCVI Международной научно-практической конференции «Молодой исследователь: вызовы и перспективы».

Глава 1 Анализ предметной области

Необходимо решить задачу прогнозирования трудоемкости разработки программного обеспечения. Для решения данной задачи собираются и обрабатываются данные для обучения классических алгоритмов машинного обучения, ансамблевых моделей и нейронных сетей, затем выделяются признаки, оцениваются результаты, полученные с их использованием. На основании результатов определяется лучший из алгоритмов. Этот алгоритм будет использоваться в качестве модуля системы анализа в разрабатываемой веб системы для предсказания трудоемкости разработки программного обучения.

Под трудоемкостью подразумевается затраты труда рабочего времени на производство единицы продукции. Она характеризует время, необходимое специалисту и коллективу для создания программного продукта. Для определения трудоемкости разработки в первую очередь составляется перечень этапов и видов работ, необходимых для выполнения. Особое внимание уделяется логическому упорядочению последовательности выполнения отдельных видов работ.

При составлении оценки трудоемкости разрабатываемого продукта учитываются следующие этапы:

– выбранная методология разработки,
– проектирование структуры базы данных. Выбирается наиболее подходящая модель хранения и представления данных,
– разработка программного продукта. Осуществляется программная реализация основных компонент программы,
– создание сопровождающей технической и пользовательской документации,
– отладка программного продукта. Отладка происходит в процессе
– разработки продукта, а также после ее завершения,

– тестирование программного продукта. Проводится проверка продукта на наличие в нем ошибок.

1.1 Обзор существующих работ

Был проведен обзор существующих алгоритмов для решения задачи оценки трудоемкости разработки программного обеспечения. Результат обзора представлен в таблице 1.

Таблица 1 – Обзор существующих алгоритмов

Авторы	**Название статьи**	**Используемые методы**
Ahmed Banimustafa	Predicting Software Effort Estimation Using Machine Learning Techniques [7]	Naïve Bayes, Logistic Regression, Random Forest
Shashank Mouli Satapathy	Effort Estimation Methods in Software Development using Machine Learning Algorithms [8]	Decision Tree, Stochastic Gradient Boosting, Random Forest, Support Vector Regression
Mohammed Zubair Khan, Omar H. Alhazmi	Software Effort Prediction Using Ensemble Learning Methods [9]	Ensemble learning bagging with base learner Linear regression, SMOReg, MLP, random forest, REPTree, and M5Rule
Carvalho, Halcyon & Lima, Marília & Santos, Wylliams & Fagundes, Roberta.	Ensemble Regression Models for Software Development Effort Estimation: A Comparative Study. [10]	Bagging, Stacking ensemble models
Prof. A. J. Singh, Mukesh Kumar	Comparative Study On Effort Estimation Using Different Data Mining Techniques [11]	Naive-Bayes, Random Forest, Artificial Neural Network, Support Vector Machine

Продолжение таблицы 1

Авторы	**Название статьи**	**Используемые методы**
Prof. A. J. Singh, Mukesh Kumar	Comparative Analysis on Prediction of Software Effort Estimation Using Machine Learning Techniques [12]	Linear Regression, Multi-layer perceptron, Random Forest,
Nassif, Ali & Azzeh, Mohammad & Capretz, Luiz & Ho, Danny	Neural Network Models for Software Development Effort Estimation: A Comparative Study [13]	Multilayer Perceptron, General Regression Neural Network, Radial Basis Function Neural Network, Cascade Correlation Neural Network
A. G. Priya Varshin, K. Anitha Kumari	Predictive analytics approaches for software effort estimation [14]	Multilinear Regression, Random Forest, Support Vector Regression, Decision Tree, Neuralnet, ElasticNet Regression, Lasso Regression

В первой статье [7] использовали набор данных, первоначально собранный NASA. Провели анализ датасета с использованием следующих алгоритмов: Naïve Bayes, Logistic Regression, Random Forests. В результате все три метода машинного обучения оказались успешными в прогнозировании большинства классов трудозатрат проекта с довольно хорошими показателями производительности, и все они превзошли прогноз модели COCOMO. Random Forests показал лучшую точность (93%) по сравнению с другими применяемыми методами.

Во второй статье [8] авторы провели анализ датасета ISBSG Software Project Data с использованием следующих алгоритмов: Decision Tree Technique, Stochastic Gradient Boosting Technique, Random Forest Technique, Support Vector Regression Technique. В результате как для новых, так и для усовершенствованных веб-проектов метод Support Vector Regression Technique RBF показывает лучшие результаты, чем другие методы машинного обучения, поскольку он обеспечивает минимальную ошибку и более высокую точность прогнозирования для шести типов рассматриваемых проектов.

В третьей статье [9] авторы провели анализ набора данных, основанных на 499 проектах, известных как China. Также был реализован алгоритм выбора признаков, чтобы изучить влияние алгоритма выбора признаков BestFit и генетического алгоритма, с использованием ансамблевого обучения с алгоритмами Linear regression, SMOReg, MLP, random forest, REPTree, M5Rule. Результаты показали, что средняя величина относительной ошибки правила Bagging M5 rule с генетическим алгоритмом в качестве выбора признаков составляет 10%, что делает его лучше, чем другие алгоритмы.

В четвертой статье [10] рассматривалось восемь различных ансамблевых моделей для оценки усилий разработки программного обеспечения, сравнивались друг с другом на основе точности прогноза по критерию среднего абсолютного остатка (MAR). Использовались два типа ансамбля – беггинг и стекинг – с различными алгоритмами: линейная регрессия, робастная регрессия, ридж-регрессия, регрессия лассо. Результаты показали, что предлагаемые ансамблевые модели, помимо обеспечения высокой эффективности по сравнению с их аналогами, дают наилучшие ответы для оценки трудозатрат в программном проекте.

В пятой статье [11] авторы использовали студенческие наборы данных с количеством записей от 100 до 600 и сравнивали оценки усилий, необходимых для разработки программного приложения следующих

алгоритмов: Naive-Bayes, Random Forest, Artificial Neural Network, Support Vector Machine. Средняя оценка усилий с помощью алгоритма ANN составила 39,148, что оказалось лучше, чем у NB (50,67), RF (51,87) и SVM (44,61).

В шестой статье [12] рассматривались такие модели машинного обучения, как Linear Regression, Multi-layer perceptron, Random Forest. Источником данных служил датасет Desharnais. Датасет был предварительно обработан и исключены атрибуты, из двенадцати изначальных остались семь. По результатам наилучшим стал метод Linear Regression, как до исключения атрибутов, так и после. Показатель MAE для Linear Regression равен 2013.79, показатель RMSE равен 2824.57.

В седьмой статье [13] исследовали четыре разные модели нейронных сетей - многослойный персептрон, нейронную сеть с общей регрессией, нейронную сеть с радиальной базисной функцией и нейронную сеть с каскадной корреляцией. Сравнивали точности прогноза, основанной на критерии средней абсолютной ошибки, исследовали склонность модели к переоценке или недооценке, и как каждая модель классифицирует важность своих входных данных. Для обучения и проверки использовался набор данных от Международной группы стандартов сравнительного анализа программного обеспечения (ISBSG). Основной набор данных ISBSG был отфильтрован, а затем разделен на пять наборов данных в зависимости от производительности каждого проекта. Результаты показали, что четыре модели имеют тенденцию к завышению оценок в 80% наборов данных, а значимость входных данных модели зависит от выбранной модели. Кроме того, нейронная сеть с радиальной базисной функцией превосходит остальные три модели в большинстве наборов данных, построенных по критерию среднего абсолютного остатка.

В восьмой статье [14] модели машинного обучения проверялись на нескольких датасетах: Desharnais, Maxwell, China. Применялись следующие

методы: Multilinear Machines, Random Forest, Support Vector Regression, Decision Tree, Neuralnet, ElasticNet Regression, Lasso Regression. Для датасета Desharnais наилучшую работу показал метод Support Vector Regression (MAE = 1888.018, RMSE=2361.356, R2 =-0.02994683). Для датасета Maxwell наилучшую работу показал метод ElasticNet Regression (MAE=3113.22, RMSE=4536.323, R2=0.7324483). Для датасета China наилучшую работу показал метод Lasso Regression (MAE=330.683, RMSE=541.6755, R2=0.9927162).

В результате проведенного анализа существующих работ для моей системы были выбраны следующие алгоритмы: Support Vector Regression, Random Forest, Artificial Neural Network, General Regression Neural Network, Radial Basis Function Neural Network. Так же будет рассмотрен ансамбль методов стекинг.

1.2 Аналитический обзор существующих систем

Из анализа области исследования можно сделать вывод, что есть место дальнейшим исследованиям и улучшениям. В качестве направления исследования было выбрано применение методов машинного обучения для оценки трудоемкости разработки ПО. Для полного анализа были рассмотрены существующие инструменты для решения оценки трудоемкости.

1.2.1 Jira

JIRA - это коммерческая система, разработанная австралийской компанией Atlassian [15]. Используется для управления проектами, отслеживает ошибок и проблемы, связанные с программным обеспечением и мобильными приложениями. Имеет веб-интерфейс и интеграцию со сторонними инструментами разработки, такими как Git, Slack и другие.

JIRA решает задачу, связанную с трудоемкостью посредствам настраиваемых полей с первичной и фактической оценкой, а

также предоставляя ряд отчетов, которые показывают статистику для конкретных проектов.

В таблице есть четыре следующих поля учета времени:

– “Исходная оценка” - исходная оценка общего количества времени, которое потребуется для завершения этого вопроса,
– “Расчетное оставшееся время” - текущая оценка оставшегося времени, необходимого для решения этой проблемы,
– “Затраченное время” - время, потраченное на решение проблемы. Это совокупное количество времени, которое было зарегистрировано для решения этой проблемы,
– “Точность” - точность исходной оценки по сравнению с текущей оценкой проблемы. Это разница между суммой полей «Затраченное время» и «Расчетное оставшееся время» и полем «Исходная оценка».

Для более расширенного отображения прогресса по задачам применяется индикатор “Work Ratio”. Он показывает информацию об учете времени в виде процента выполнения. Выражается в процентах и вычисляется по формуле:

workRatio = (потраченное фактическое время / изначальная оценка) x 100

По умолчанию Jira не отображает данный индикатор. Он помогает увидеть перерасход времени по задаче, а, следовательно, сделать вывод о некорректной оценке трудозатрат на нее. Кроме того, есть процент, который говорит нам, где мы находимся с точки зрения завершения. WorkRatio коэффициент поможет понять текущее состояние задачи и качество оценки.

1.2.2 GoodDay

GoodDay - это облачное решение для управления проектами для разных по масштабу предприятий [16].

GoodDay решает задачу, связанную с трудоемкостью посредствам таких инструментов, как доска Kanban для одноименного метода управления разработкой, диаграмма Ганта для планирования этапов разработки, модуль GoodDay Business Intelligence для отслеживает и выполняет анализ времени пользователей, затраченных на задачу.

Модуль GoodDay Business Intelligence автоматически отслеживает и анализирует всю активность и может автоматически оценивать усилия, затраченные на задачу. Функция основана на том, что система знает продолжительность рабочего дня каждого пользователя и имеет все данные обо всех повседневных действиях. В основе модуля лежит самообучающийся алгоритм. Система анализирует все прошлые действия и оценивает, что может потребоваться пользователю для выполнения задач. Автоматические оценки в начале работы с программой могут быть неточными, поскольку они основаны на активности за один день, с течением времени модуль становится более точными.

1.2.3 COCOMO II Web

Среди бесплатных инструментов оценки проекта есть COCOMO II Web Portal от Center for Systems and Software Engineering от Университета Южной Калифорнии [17]. В основе модуля анализатора в этом инструменте лежит методика COCOMO. Методика COCOMO является алгоритмической моделью оценки стоимости разработки программного обеспечения, которая позволяет оценивать трудоемкость и время разработки программного продукта. Для расчета необходимо ввести обязательные параметры. На выходе - текстовый файл.

1.2.4 SLIM-Estimate

Компания Quantitative Software Management (QSM) предлагает инструмент оценки проектов SLIM-Estimate [18]. SLIM-Estimate может дать оценку стоимости и времени для заданного набора требований. Также

предлагает стратегии для разработки и реализации проекта. Помимо оценки стоимости программного обеспечения, он позволяет использовать множество различных процессов проектирования, используемых разработчиками, включая Agile-разработку, бизнес-аналитику, проектирование и архитектурный дизайн, сервис-ориентированную архитектуру. Основной минус этого продукта в том, что он платный.

Итоговое сравнение инструментов представлено в таблице 2.

Таблица 2 - Сравнение существующих систем

Название продукта	Свободный доступ	Сбор данных по уже выполненным программным продуктам	Предсказание временных затрат на выполнение нового программного продукта на основе проанализированной информации	Доступность из сети Интернет
Jira	-	+	-	+
GoodDay	-	+	-	+
COCOMO II Web Portal	+	-	-	+
SLIM-Estimate	-	+	-	+

1.3 Вывод

Совершенствование решений в данной области возможно за счет изменения глубины анализа, использования новых методов аналитики. Наиболее часто применяются модели: регрессия опорных векторов, случайный лес и нейронные сети.

Глава 2 Анализ существующих методов и подходов

2.1 Обзор существующих математических моделей для решения задачи трудоемкости разработки корпоративных информационных систем

2.1.1 Регрессия опорных векторов

Support Vector Regression (SVR) - алгоритм регрессии SVM (Support Vector Machine). Это алгоритм обучения с учителем, который используется для прогнозирования дискретных значений.

В машинном обучении SVM - это модели обучения с учителем со связанными алгоритмами обучения, которые анализируют данные, используемые для классификации и регрессионного анализа. В регрессии опорных векторов прямая линия, которая требуется для соответствия данным, называется гиперплоскостью.

Задача алгоритма SVM - найти гиперплоскость в n-мерном пространстве, которая четко классифицирует точки данных. Точки данных по обе стороны от гиперплоскости, которые ближе всего к гиперплоскости, называются опорными векторами. Они влияют на положение и ориентацию гиперплоскости и, таким образом, помогают построить SVM.

Версия SVM для регрессии была предложена в 1996 году Владимиром Н. Вапником, Харрисом Друкером, Кристофером Дж. К. Берджесом, Линдой Кауфман и Александром Дж. Смолой [19]. В регрессии опорных векторов используется тот же принцип, что и в SVM. В SVR наиболее подходящей линией является гиперплоскость с максимальным количеством точек. Он пытается подобрать лучшую линию в пределах порогового значения. Пороговое значение - это расстояние между гиперплоскостью и линией границы.

Рассмотрим различные гиперпараметры, которые используются в регрессии опорных векторов.

1. Гиперплоскость:

Гиперплоскости - это границы решений, которые используются для прогнозирования непрерывного вывода. Точки данных по обе стороны от гиперплоскости, которые ближе всего к гиперплоскости, называются опорными векторами. Они используются для построения необходимой линии, которая показывает прогнозируемый результат алгоритма.

2. Ядро:

Ядро - это набор математических функций, которые принимают данные на вход и преобразуют их в требуемую форму. Обычно они используются для поиска гиперплоскости в пространстве более высоких измерений.

Наиболее широко используемые ядра включают линейные, нелинейные, полиномиальные, радиальные базисные функции (RBF) и сигмовидные. По умолчанию в качестве ядра используется RBF. Каждое из этих ядер используется в зависимости от набора данных.

3. Граничные линии:

Это две линии, проведенные вокруг гиперплоскости на расстоянии ε (эпсилон). Он используется для создания поля между точками данных.

Модель SVR зависит только от подмножества обучающих данных, так как функция стоимости построения модели игнорирует любые обучающие данные, близкие к предсказанию модели.

Достоинства SVR:

– вычислительная сложность не зависит от размерности входного пространства,
– обладает отличными возможностями обобщения с высокой точностью прогнозирования.

Недостатки SVR:

– не подходят для больших наборов данных,
– в случаях, когда количество функций для каждой точки данных превышает количество выборок обучающих данных, SVR будет работать хуже,
– модель принятия решения не очень хорошо работает, когда в наборе данных больше шума, целевые классы перекрываются.

2.1.2 Случайный лес

Суть алгоритма случайного леса (Random forest) заключается в использовании ансамбля решающих деревьев. Данный алгоритм был предложен Лео Брейманом и Адель Катлер [25]. Он сочетает в себе две основные идеи: метод бэггинга и метод случайных подпространств. Бэггинг (bagging) был предложен Л. Брейманом в 1996 году [26]. Бэггинг – способ построения композиций регрессоров, в котором они обучаются независимо друг от друга. Благодаря такому подходу метод можно сделать параллельным. Бэггинг позволяет вычислить оценки обобщающей способности, оптимизировать число алгоритмов, подобрать оптимальные параметры модели.

Случайный лес строится путем объединения прогнозов нескольких деревьев, каждое из которых обучается изолированно от других. Дерево является конечным связным графом с множеством вершин V. Эти вершины не содержат циклов и имеют одну выделенную вершину, которая называется корнем дерева, в него не входит ни одно ребро. Вершина, которая не имеет выходящих ребер, называется листом, соответствуют результату регрессии. Остальные вершины называются внутренними, соответствуют признакам. Выходящие ребра связывают каждую внутреннюю вершину v с левой (Lv) и с правой (Rv) дочерними вершинами. Дерево решений - это процесс

принятия решений с учетом возможных альтернатив на основе признаков исследуемого объекта.

Достоинства модели случайного леса:

– эффективно обрабатывает данных с большим числом признаков,
– высокое качество получаемых моделей, сравнимое с SVM и бустингом, и в некоторых случаях лучше, чем у нейронных сетей [27];
– высокая параллелизуемость и масштабируемость,
– не чувствителен к масштабированию значений признаков,
– хорошо обрабатывает непрерывные и дискретные признаки,
– хорошие свойства параллелизуемости и масштабируемости,
– эффективен на большом спектре задач.

Недостатки модели случайного леса:

– плохо выявляет скрытые закономерности,
– плохо обрабатывает категориальные признаки,
– медленная скорость обработки пропущенных значений.
– необходим большой объема памяти для хранения модели,
– склонен к переобучению на зашумленных задачах [28],
– большой размер структуры данных.

2.1.3 Искусственная нейронная сеть

Искусственными нейронными сетями (Artificial Neural Networks) называются вычислительные модели, созданные на основе на модели нейрона человека. Человеческий мозг состоит из миллионов нейронов. Он отправляет и обрабатывает сигналы в виде электрических и химических сигналов. Эти нейроны связаны с особой структурой, известной как синапсы, которые позволяют нейронам передавать сигналы. Нейронные сети формируются из большого количества смоделированных нейронов.

Нейронную сеть можно представить в виде ориентированного графа. Он состоит из узлов – аналогов нейронов, которые соединены дугами - аналоги дендритов и синапсов. Каждая дуга связана с весом в каждом узле. Примените значения, полученные в качестве входных данных узлом, и определите функцию активации вдоль входящих дуг, скорректированную весами дуг.

Нейронная сеть применяется как для задач классификации, так и для задач регрессии непрерывных целевых атрибутов.

Искусственная нейронная сеть организовывается по слоям. Слои состоят из множества взаимосвязанных «узлов», которые содержат «функцию активации». Нейронная сеть может содержать следующие три слоя:

Входной слой - шаблоны в сеть, которая сообщается с одним или несколькими «скрытыми слоями». Количество входных узлов во входном слое равно количеству атрибутов. Узлы входного слоя не изменяют данные. Они получают значение на входе и дублируют это значение на множество выходов, передавая их скрытым узлам.

Скрытый слой определяет активности каждого скрытого объекта. В скрытом слое обработка осуществляется через систему взвешенных «связей». Нейронная сеть может содержать один или несколько скрытых слоев. Скрытые слои применяют заданные преобразования к входным значениям внутри сети. Значения, которые попадают в скрытый узел, умножаются на веса. Затем взвешенные входные данные складываются для получения одного числа.

Выходной слой получает соединения от скрытых слоев. Он возвращает выходное значение, соответствующее предсказанию переменной ответа. В задачах регрессии выходной слой состоит из одного узла.

Иллюстрация представлена на рис. 1.

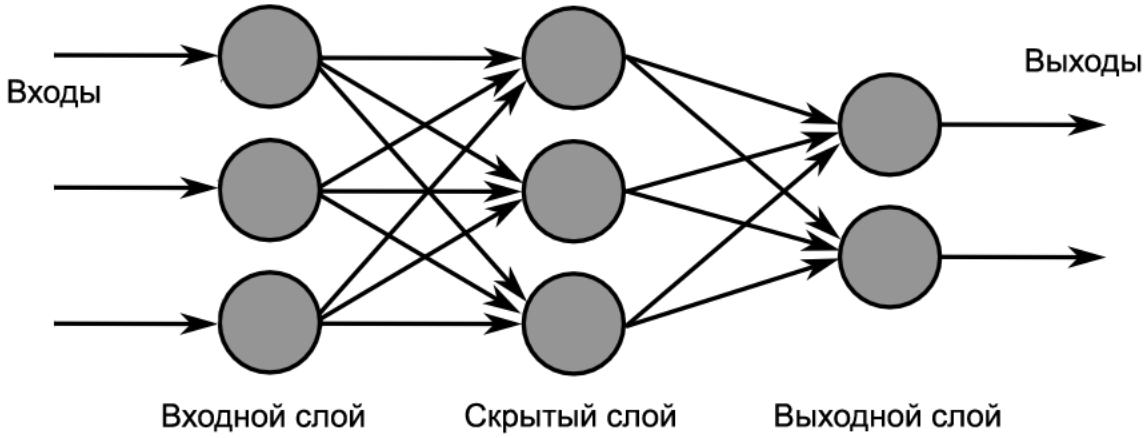

Рис. 1. Искусственная нейронная сеть

Преимущества искусственной нейронной сети:

– хорошо работают с линейными и нелинейными данными,
– решение задач с неизвестными закономерностями и зависимостями между входными и выходными данными,
– устойчивость к шумам во входных данных,
– быстродействие за счет использования параллельной обработки информации,
– не требует перепрограммирования,
– отказоустойчивость.

Недостатки нейронных сетей

– требуют большого разнообразия обучения для реальной работы,
– дают очень мало информации о том, что эти модели на самом деле делают.

2.1.4 Сеть радиальных базисных функций

Сеть радиальных базисных функций (RBFN) - это искусственная нейронная сеть, которая в качестве функций активации применяет радиальные базисные функции. Впервые введены в 1985 г. M.J.D. Пауэлл для использования в интерполяции с несколькими переменными и приближении функций [29].

Выходом сети является линейная комбинация радиальных базисных функций входов и параметров нейрона. Данные сети имеют множество применений, такие как: аппроксимация функций, прогнозирование временных рядов, классификация и управление системой.

φ_c - это функция с симметричным выходом вокруг центра μ_c

$$\varphi_c(\bar{x}) = \varphi(\|\bar{x} - \bar{\mu}_c\|), \qquad (1)$$

где || || - векторная норма.

Чаще всего в качестве симметричной функции используется функция Гаусса:

$$y(\bar{x}) = \sum_{j=1}^{M} w_j \varphi(\|\bar{X} - \bar{\mu}_j\|), \qquad (2)$$

где w_j - веса, которые могут использоваться для представления широкого класса функций.

Преимущества RBFN:

– простая структура сети,
– используют в своей структуре один промежуточный слой, избавляя от необходимости решать вопрос о числе слоев,
– хорошие возможности аппроксимации,
– быстрая скорость обучения.

Недостатки RBFN:

– плохие экстраполирующие свойства,
– громоздки при большой размерности вектора входов.

2.1.5 Обобщенная регрессионная нейронная сеть

Обобщенные регрессионные нейронные сети (GRNN) прогнозируют, интерполируют и проводят регрессионный анализ. Полезны, когда взаимосвязь между зависимыми и независимыми переменными неизвестна и сложна. Он поддерживает как линейные, так и нелинейные отношения.

Математическая формулировка для реализации GRNN похожа на функцию распределения вероятностей. Функция вывода GRNN:

$$\acute{Y} = f(x) = \frac{\sum_{i=1}^{n} Y^i exp^{\left(-D_i^2/2\sigma^2\right)}}{\sum_{i=1}^{n} exp^{\left(-D_i^2/2\sigma^2\right)}} \qquad (3)$$

где $\acute{Y}$ - оценочное значение зависимой переменной, Y^i- значение зависимой переменной в известных местоположениях , D_i - скалярный член, учитывающий различия между точкой прогноза и обучающей выборкой для всех независимых переменных и вычисляется как

$$D_i^2 = \left(X - X^i\right)^T \left(X - X^i\right) \qquad (4)$$

Расстояние между точкой прогнозирования и обучающей выборкой определяет влияние обучающей выборки на вычисление Ý.

Если это расстояние мало, член $exp^{\left(-D_i^2/2\sigma^2\right)}$ увеличивается и равен единице при нулевой разнице. Большее значение этого члена означает, что известное значение зависимой переменной в этой обучающей выборке будет иметь большее влияние при вычислении зависимой переменной в точке прогнозирования.

Если расстояние велико, значение члена $exp^{\left(-D_i^2/2\sigma^2\right)}$ уменьшается, стремясь к нулю на больших расстояниях. Такие точки выборки не будут вносить вклад в оценку зависимой переменной в месте прогноза. Прогнозируемый результат ограничен между максимальным и минимальным известны ми значениями зависимой переменной.

GRNN имеют следующие четыре слоя нейронов:

– входной слой,

– шаблонный слой,

– слой суммирования,

– выходной слой.

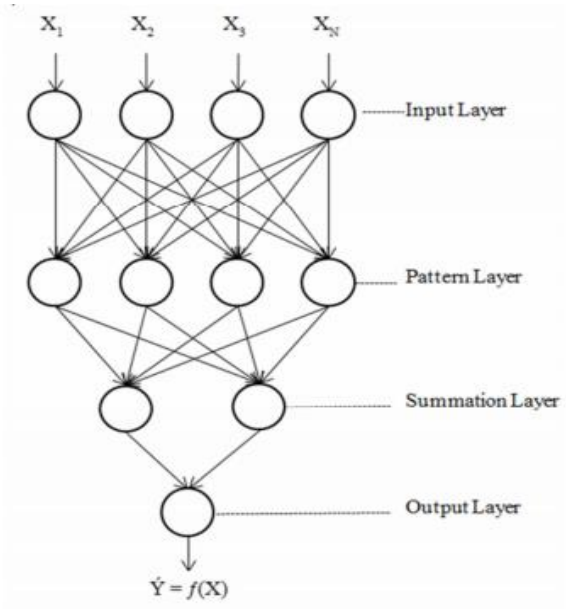

Рис. 2. Общая структура GRNN

На рис. 2 показана общая структура GRNN с четырьмя слоями [30]. GRNN может аппроксимировать функцию и оценить значение зависимой переменной из набора независимых переменных. Во входном слое содержится столько нейронов, сколько переменных во входном наборе данных. Каждая точка входных данных затем сохраняется в слое образца. Количество нейронов в слое шаблона равно общему количеству точек данных. Значение зависимой переменной (Y) в точке прогноза вычисляется как разница между значениями независимых переменных в точке прогноза и их соответствующими значениями в других точках, в которых независимые переменные известны. Слой суммирования вычисляет члены числителя и знаменателя для уравнения (3), используя коэффициент разности независимых переменных (в известном и неизвестном местоположении) и зависимой переменной (в известном местоположении). Последний слой называется выходным слоем.

Преимущества GRNN:

– однопроходное обучение, поэтому обратное распространение не требуется,
– высокая точность оценки благодаря использованию функций Гаусса,

– может обрабатывать шумы на входах,
– требует меньшего количества наборов данных.

Недостатки GRNN:

– его размер может быть огромным, что делает его затратным в вычислительном отношении,
– оптимального метода его улучшения не существует.

2.1.6 Ансамбль

Ансамблевое обучение - это совместное использование алгоритмов машинного обучения для решения задач классификации и регрессии. Цель ансамблевых методов заключается в объединении прогнозов нескольких оценок, полученных с заданным алгоритмом обучения для получения лучшей эффективности по сравнению с получением одной оценки. Существуют разные типы методов ансамблевого обучения, которые различаются в основном типом используемых моделей (однородные или гетерогенные модели), выборкой данных (с заменой или без нее, k-кратной и т. Д.) И функцией принятия решения (голосование, среднее, мета модель и т. д.). В данной работе рассмотрим тип ансамбля стекинг.

Стекинг (Stacking Generalization) - это метод ансамблевого обучения, введеный Дэвидом Х. Вольпертом в 1992 году [31], позволяющий комбинировать несколько регрессионных моделей с помощью мета-регрессора. Принцип работы изображен на рис. 3. Алгоритмы, находящиеся на базовом уровне, обучаются на основе обучающих данных, а их результирующая мета модель обучается на конечных результатах всей модели базового уровня как функции.

Рис. 3. Стекинг

Преимущества стекинга:

– уменьшение разброса среднего навыка прогнозной модели,
– повышение средней производительности прогнозирования по любому участвующему члену в ансамбле,
– может использовать возможности ряда хорошо работающих моделей для задач классификации и регрессии, и делать прогнозы, которые имеют лучшую производительность, чем любая отдельная модель в ансамбле.

Недостатки стекинга:

– улучшает результаты базовых классификаторов при относительно большом числе обучающих примеров.

2.2 Вывод

В данной главе был проведен обзор и сравнение существующих математических моделей для решения задачи трудоемкости разработки программного обеспечения корпоративных систем. Данные методы выбраны для проведения эксперимента над данными и в качестве анализаторов для будущей системы.

Глава 3 Проектирование и реализация модели

3.1 Проектирование системы

3.1.1 Составление требований к системе

Система для оценки трудоемкости разработки программного обеспечения корпоративных информационных систем с использованием методов машинного обучения предоставляет модель для прогнозирования трудоемкости затрат на исполнение программного продукта

Система предназначена обрабатывать данные по уже выполненным продуктам и на основе обученной модели предоставлять оценку временных затрат по новому программному продукту.

3.1.2 Функциональные требования к системе

Разрабатываемая система для оценки трудоемкости разработки программного обеспечения корпоративных информационных систем с использованием методов машинного должна удовлетворять следующим функциональным требованиям:

1. система должна осуществлять сбор данных по уже выполненным программным продуктам на основе сформированной базы данных,
2. система должна предсказывать временные затраты на выполнение нового программного продукта на основе проанализированной информации.

3.1.3 Нефункциональные требования к системе

Разрабатываемая система ориентирована на пользователей, географически удаленных друг от друга. Значит она должна быть доступна из сети Интернет. Все страницы веб-интерфейса должны корректно отображаться в следующих наиболее популярных браузерах: Google Chrome версии 33 и выше, Mozilla Firefox версии 29.0 и выше. А так же мобильные версии данных браузеров.

3.1.4 Варианты использования системы

В ходе выполнения диссертации была разработана диаграмма вариантов использования системы для оценки трудоемкости разработки программного обеспечения корпоративных информационных систем с использованием методов машинного обучения, изображенная на рис. 4. Диаграмма включает в себя следующих основных актеров, которые взаимодействуют с системой:

1. пользователь использует систему получения оценки трудоемкости,
2. система анализа обеспечивает автоматизированное получение данных, которые обрабатывает веб-система.

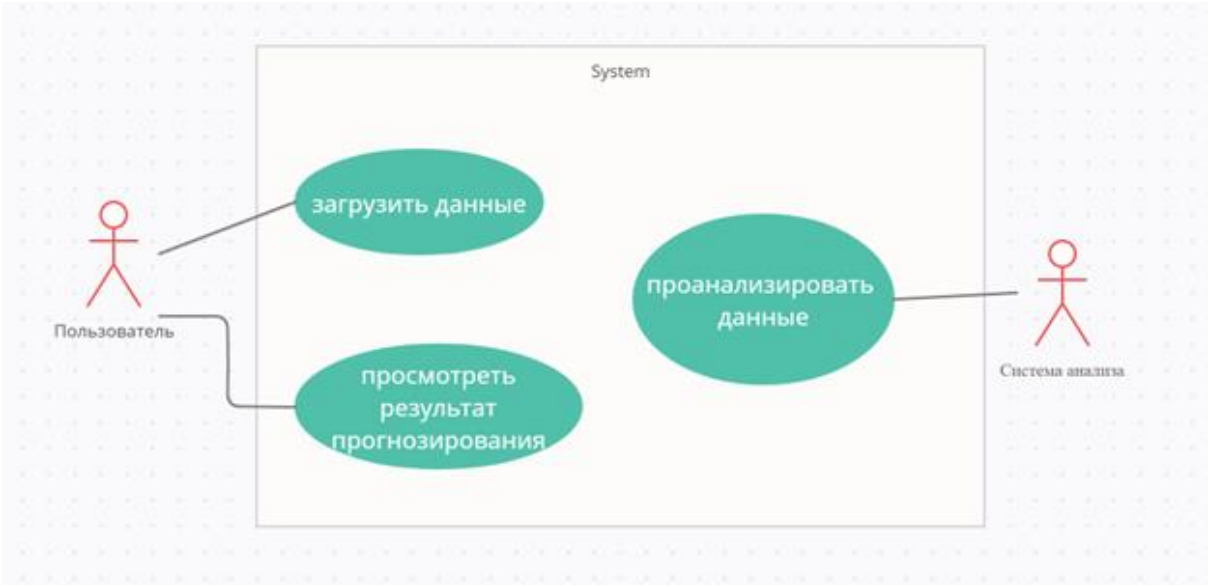

Рис. 4. Диаграмма вариантов использования

Основные варианты использования веб-системы для оценки трудоемкости разработки программного обеспечения корпоративных информационных систем.

Пользователь:

– «загрузить данные»: пользователь может загрузить данные для обучения анализатора и получению прогноза оценки трудоемкости,

– «просмотреть результат прогнозирования»: пользователь может просмотреть результат прогнозирования системы на основе обработанных данных.

Блок-схема работы анализатора при создании новой модели представлена на рис.5.

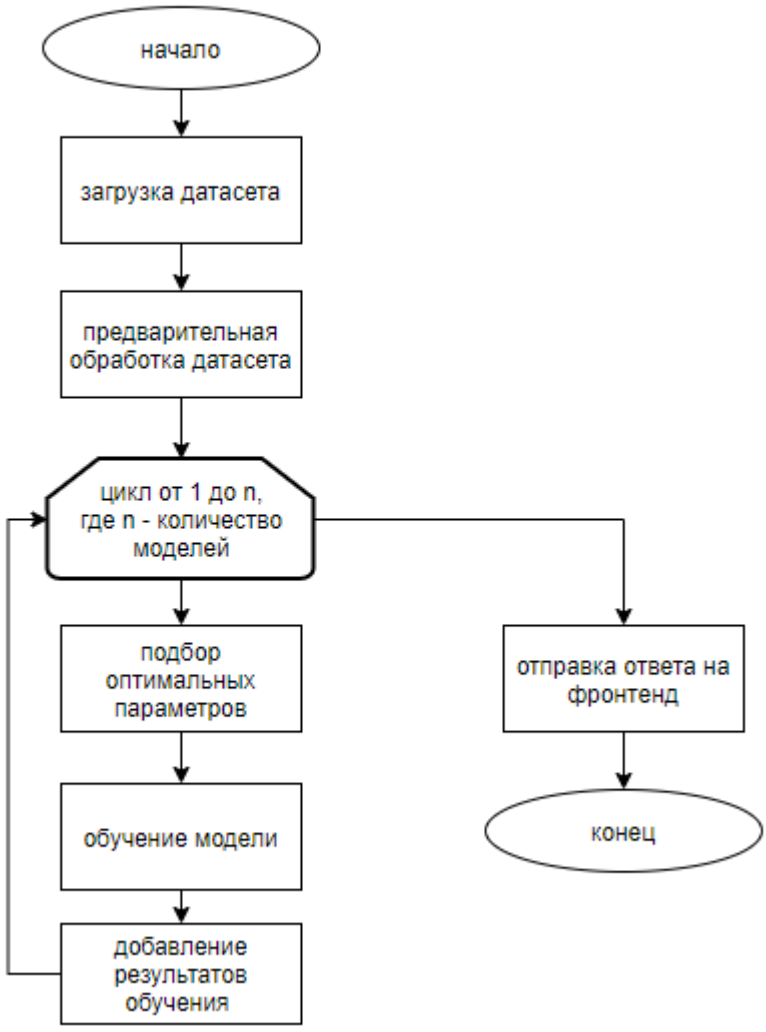

Рис. 5. Блок схема работы анализатора

3.2 Проектирование системы

3.2.1 Компоненты системы

На рис. 6 представлена диаграмма компонентов системы анализа трудоемкости разработки программного обеспечения.

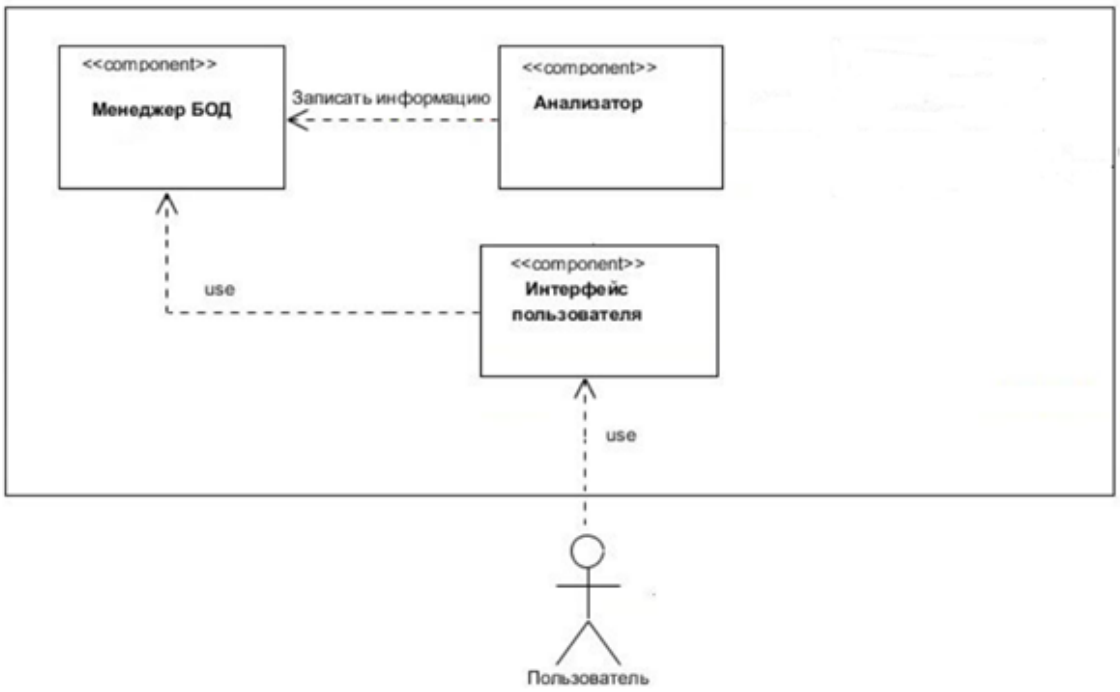

Рис. 6. Диаграмма компонентов системы

Диаграмма демонстрирует программную систему, разделенную на структурные компоненты и зависимости между компонентами. Система включает в следующие компоненты:

– интерфейс клиента – внутренний компонент, предназначен для взаимодействия пользователя с системой,
– менеджер БОД – внутренний компонент, отвечает за взаимодействие с базой данных, а также хранит обработанную анализатором информацию,
– анализатор – внешний компонент, обеспечивает обработку, анализ данных и получение оценок.

На рис. 7 представлена диаграмма последовательности для сценария взаимодействия пользователя с системой.

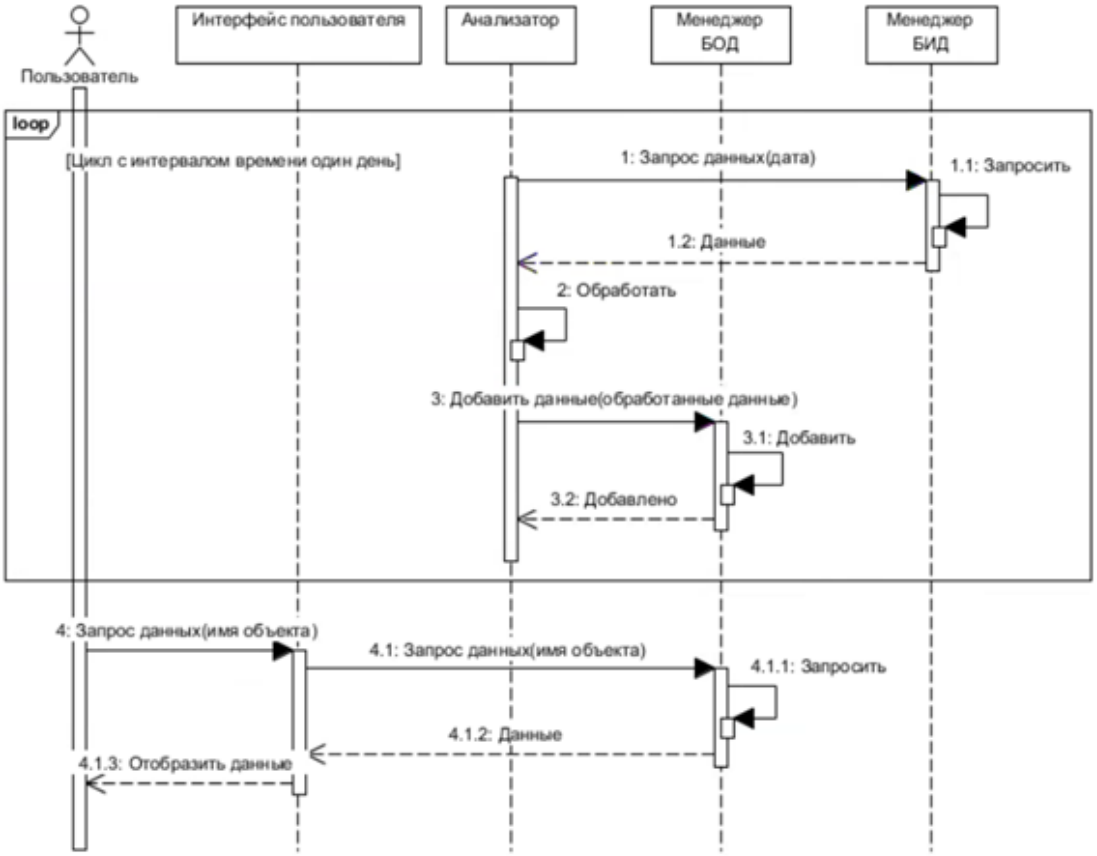

Рис. 7. Диаграмма последовательности

Компонент «Анализатор» запрашивает данные для анализа у «Менеджера БИД». После получения данных анализатор обрабатывает их. Результаты полученного анализа сохраняются в базе данных с помощью компонента «Менеджер БОД». При запросе данных пользователь обращается к пользовательскому интерфейсу. Пользовательский интерфейс запрашивает данные у «Менеджер БОД» и отображает результат пользователю.

3.2.2 Схема базы данных

База данных является важным компонентом разрабатываемой системы. Она предназначена для хранения информации, обработка которой составляет ключевую часть работы системы.

В системе будет использоваться СУБД, схема которой представлена на рис. 8.

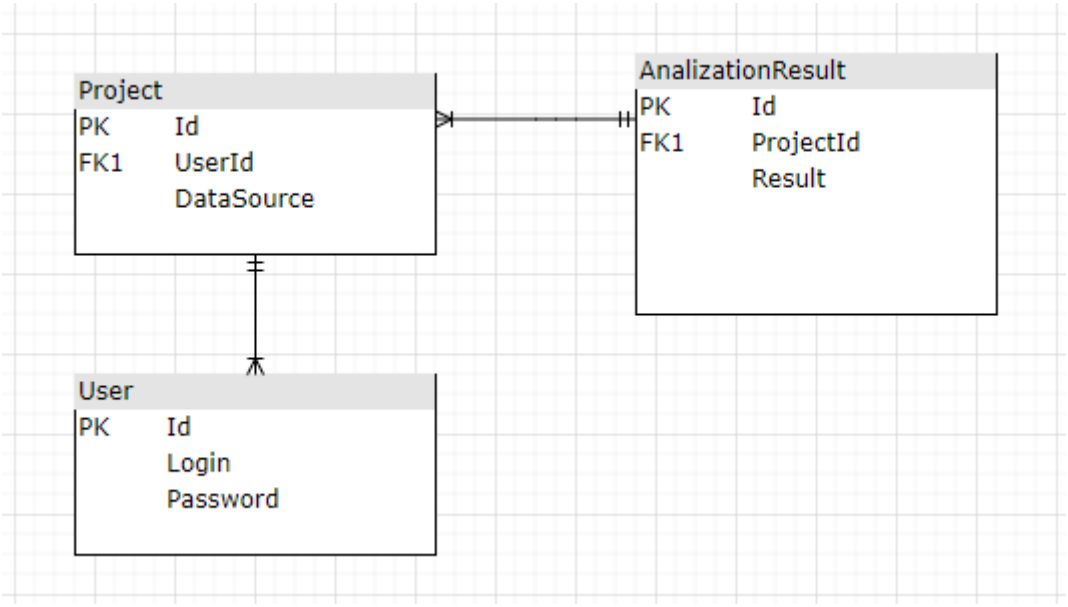

Рис. 8. Схема базы данных

Таблица User содержит информацию о пользователях системы, необходимой для отображения проектов и процедуры авторизации в системе. Таблица связана с таблицей проектов (Project) отношением «один-ко-многим».

Таблица 3 - Таблица User

Название поля	**Тип поля**	**Семантика**
Id	INT NOT NULL	Первичный ключ
Login	VARCHAR(255) NOT NULL	Логин
Password	VARCHAR(255) NOT NULL	Пароль

Таблица Project содержит информацию о проектах, которые анализировала система. Таблица связана с таблицей пользователей (User) отношением «много-к-одному».

Таблица 4 - Таблица Project

Название поля	**Тип поля**	**Семантика**
Id	INT NOT NULL	Первичный ключ
UserId	INT NOT NULL	Связь с таблицей User
DataSource	VARCHAR(255) NOT NULL	Ссылка на датасет данных по проекту

Таблица AnalyzationResult содержит результат анализа опредленной модели. Таблица связана с таблицей проектов (Project) отношением «один-к-одному».

Таблица 5 - Таблица AnalyzationResult

Название поля	**Тип поля**	**Семантика**
Id	INT NOT NULL	Первичный ключ
ProjectId	INT NOT NULL	Связь с таблицей Project
Result	VARCHAR(255) NOT NULL	Результат анализа

3.3 Построение модели анализатора

3.3.1 Выбор меры эффективности алгоритмов

Наиболее часто для оценки о производительности регрессионной модели; используются следующие три метрики:

1. Среднеквадратичная ошибка (RMSE)
2. Средняя абсолютная ошибка (MAE)

3. Коэффициент детерминации

Средняя абсолютная ошибка

Средняя абсолютная ошибка (Mean Absolute Error, MAE) - это среднее значение разницы между исходными и прогнозируемыми значениями. Он дает представление о том, как далеко прогнозы от фактического результата. Но они не дают представления о том, занижаем ли мы данные или переоцениваем их. Математически ошибка вычисляется как:

$$MAE = \frac{1}{N}\sum_{j=1}^{N}\left|y_j - \hat{y}_j\right| \quad (3)$$

Чем больше MAE, тем критичнее ошибка. Единица MAE такая же, как и единица прогнозируемой переменной.

Среднеквадратичная ошибка (MSE)

Среднеквадратичная ошибка (Mean Squared Error, MSE) принимает среднее значение квадрата разницы между исходными значениями и предсказанными значениями.

$$MSE = \frac{1}{N}\sum_{j=1}^{N}\left(y_j - \hat{y}_j\right)^2 \quad (4)$$

Это также важная функция потерь для алгоритмов, подходящих или оптимизированных с использованием метода наименьших квадратов для задачи регрессии. Здесь «наименьшие квадраты» относятся к минимизации среднеквадратичной ошибки между предсказаниями и ожидаемыми значениями.

Коэффициент детерминации

Коэффициент детерминации или R2 представляет собой долю дисперсии в зависимой переменной, которая объясняется моделью линейной регрессии. Это оценка без шкалы, т.е. независимо от того, малые или большие значения, значение квадрата R2 будет меньше единицы.

$$R^2 = 1 - \frac{\sum_{j=1}^{N}(y_i - \hat{y})^2}{\sum_{j=1}^{N}(y_i - \bar{y})^2} \quad (5)$$

Оценка R2 измеряет, насколько хорошо фактические результаты воспроизводятся линией регрессии. Это поможет вам понять, насколько хорошо независимая переменная скорректирована с дисперсией в вашей модели. Это означает, насколько хороша ваша модель для набора данных.

Наилучшая возможная оценка - 1,0, и она может быть отрицательной (модель может быть произвольно хуже). Чем ближе его значение к единице, тем лучше ваша модель. Это связано с тем, что линия регрессии хорошо соответствует набору данных, либо точки данных распределены с низкой дисперсией. Что уменьшает значение суммы остатков. Постоянная модель, которая всегда предсказывает ожидаемое значение y, игнорируя входные характеристики, получит оценку R2 равную нулю.

3.3.2 Описание данных

В данной работе использовались следующие наборы данных: Desharnais, SiP Dataset и MaxwellDevelopment. Рассмотрим каждый из них.

Desharnais

Это общедоступный набор данных PROMISE Software Engineering Repository, для воспроизведения, проверки, опровержения и улучшения прогнозных моделей программной инженерии [32]. Датасет состоит из 81 записи и 10 атрибутов, описание которых представлено в таблице 6.

Таблица 6 - Атрибуты датасета Desharnais

Название атрибута	**Описание**	**Тип данных**
id	идентификатор записи	integer
Project	идентификатор проекта	integer
TeamExp	опыт команды (в годах)	integer
ManagerExp	опыт управления (в годах)	integer

Продолжение таблицы 6

Название атрибута	**Описание**	**Тип данных**
YearEnd	год окончания проекта	integer
Length	срок реализации проекта (в месяцах)	integer
Effort	фактическая трудоемкость (в человеко-часах)	integer
Transactions	количество основных логических транзакций в системе	integer
Entities	количество сущностей в модели данных системы	integer
PointsNonAdjust	размер проекта (в функциональных точках)	integer

SiP

Анализ более десяти лет коммерческой разработки с использованием Agile (10 000 уникальных оценок задач, выполненных 22 разработчиками, в рамках 20 кодов проектов) документируется в ходе беседы с участием аналитика данных и директора компании, создавшей набор данных SiP.

Датасет состоит из 20 атрибутов и содержит 10000 записей. Описание атрибутов представлено в таблице 7.

Таблица 7 - Атрибуты датасета SiP

Название атрибута	**Описание**	**Тип данных**
TaskNumber	идентификатор задачи	int
Summary	краткое текстовое описание действия, необходимого для выполнения задачи	chr
Priority	приоритет задачи	int
RaisedByID	идентификатор человека, создавшего задачу	int
AssignedToID	идентификатор сотрудника SiP, ответственного за завершение работы	int

Продолжение таблицы 7

Название атрибута	**Описание**	**Тип данных**
AuthorisedByID	идентификатор менеджера SiP с полномочиями подписывать задачу от имени клиента	int
StatusCode	статус задачи	char
ProjectCode	рабочий поток, с которым связана задача	char
ProjectBreakdownCode	дальнейшая разбивка рабочего потока: либо на конкретных клиентов в продукте с различной клиентской базой, либо на подсистемы в продуктах, созданных исключительно для одного клиента	chr
Category	идентификатор, определяющий категорию задачи управления, эксплуатации или разработки	chr
SubCategory	конкретный тип родительской категории	chr
HoursEstimate	десятичное значение, представляющее предполагаемое количество часов на выполнение работы	num
HoursActual	десятичное значение, представляющее общее количество часов, затраченных на выполнение Задачи всеми разработчиками SiP	num
DeveloperID	уникальный идентификатор разработчика SiP, который выполнил один или несколько фактических элементов работы, определенных в задаче	int
DeveloperHoursActual	десятичное значение, представляющее количество часов, в течение которых один разработчик SiP работал над задачей	num

Продолжение таблицы 7

Название атрибута	**Описание**	**Тип данных**
TaskPerformance	десятичное значение, представляющее недогруз (+ значение) или перерасход (- значение), достигнутый всеми разработчиками SiP для завершенной задачи	num
DeveloperPerformance	десятичное значение, представляющее недогруз (+ значение) или переполнение (- значение) для одного разработчика SiP, работающего над завершенной задачей	num
EstimateOn	дата оценки задачи	date
StartedOn	дата начала выполнения задачи	date
CompletedOn	дата завершения выполнения задачи	date

Maxwell dataset

Для оценки предложенного метода используется набор данных Maxwell, состоящий из собранных в одном из крупнейших коммерческих банков Финляндии. Датасет состоит из 29 атрибутов и содержит 64 записи. Описание атрибутов представлено в таблице 8.

Таблица 8 - Атрибуты датасета Maxwell

Название атрибута	**Описание**	**Тип данных**
Id	идентификационный номер	num
size	размер приложения (в функциональных точках)	num
effort	фактическая трудоемкость (в часах)	num
duration	продолжительность проекта от спецификации до поставки (в месяцах)	num
start app	месяц / день / год начата разработки	str
app	название приложения	str
Har	аппаратная платформа	str

Продолжение таблицы 8

Название атрибута	**Описание**	**Тип данных**
dba	Архитектура СУБД	str
ifc	тип пользовательского интерфейса	str
source	где разработан пользовательский интерфейс	str
lan1 lan2 lan3 lan4	используемые языки при разработке	str
t01	насколько активно заказчик принимал участие в разработке: 1 = очень малое участие заказчика; никто не принимал участие 2 = малое участие заказчика; 3 = среднее участие заказчика; 4 = высокое участие заказчика	num
t02	Уровень использования инструментальных ресурсов и оборудования при выполнении проекта: 1 = очень низкий; постоянные недостатки в устройствах, создание тестовых сред и тестирование требовали специальных мер 2 = Низкий; совместно используемое оборудование / машинные ресурсы; задержки на некоторых этапах работы (например, компиляция и тестирование) 3 = номинальный; достаточно при разработке; рабочая станция для всех 4 = высокий; достаточно, чтобы справиться с пиками емкости (эффективность, хранение, время отклика); 5 = очень высокий	num

Продолжение таблицы 8

Название атрибута	**Описание**	**Тип данных**
t03	Наличие программного персонала во время проекта: 1 = очень низкое; значительные проблемы с доступностью ключевого персонала; множество одновременных обязанностей клиента и обслуживания; 2 = низкое; персонал участвует в некоторых других одновременных проектах и/или обязанностях по техническому обслуживанию 3 = среднее; ключевые участники участвуют только в одном стороннем проекте 4 = высокое; участники проекта задействованы практически на постоянной основе 5 = очень высокое; квалифицированный персонал, доступный при необходимости.	num
t04	Уровень и использование стандартов: 1 = очень низкий 2 = низкий 3 = номинальный 4 = высокий 5 = очень высокий	num
t05	Уровень и использование методов: 1 = очень низкий; отсутствуют современные подходы; в основном встречи; используется частными лицами 2 = низкий; используются классические концепции 3 = номинальный; используются общеизвестные методы 4 = высокий; детально интегрированы методы и охвачено большинство видов деятельности; 5 = очень высокий; методы используются на протяжении всего жизненного цикла; методы адаптированы к конкретным потребностям проекта;	num

Продолжение таблицы 8

Название атрибута	**Описание**	**Тип данных**
t06	Уровень и использование инструментов: 1 = очень низкий; минимальные инструменты: редакторы, компиляторы и инструменты тестирования 2 = Низкий; основные инструменты: интерпретаторы, редакторы, компиляторы, отладчики, базы данных и библиотеки 3 = номинальный; среда разработки, система управления базами данных (СУБД) и поддержка большинства фаз 4 = высокий; современные инструменты, такие как CASE, планировщики проектов, генераторы приложений и стандартизированные интерфейсы между фазами 5 = очень высокий; интегрированная среда CASE на протяжении всего жизненного цикла; все инструменты поддерживают друг друга	num
t07	Требования к вычислениям, вводу / выводу и пользовательскому интерфейсу: 1 = очень низкий; нет необходимости в пользовательском интерфейсе; простые базы данных 2 = Низкий; функционально понятный; нет алгоритмических задач; 3 = номинальный; функционально типичный; нормальная, стандартная база данных; нет алгоритмов 4 = высокий; обработка более требовательная; база данных большая и сложная; новые требования к пользовательским интерфейсам 5 = очень высокий; функционально и технически сложное решение; очень сложный пользовательский интерфейс; распределенные базы данных	num

Продолжение таблицы 8

Название атрибута	**Описание**	**Тип данных**
t08	Неустойчивость требований заказчика/пользователя в ходе проекта: 1 = очень низкий; нет новых функций; стандартные компоненты; 2 = Низкий; некоторые изменения в спецификациях; некоторые новые или адаптированные функции; некоторые незначительные изменения в содержании данных 3 = номинальный; больше изменений в спецификациях, но участники проекта могли с ними справиться; незначительное воздействие (<15% новых или измененных функций) 4 = высокий; некоторые важные изменения, влияющие на общую архитектуру и требующие доработки; 15-30% функций новые или измененные 5 = очень высокий; новые требования добавляются постоянно; много переделок; более 30% новых или измененных функций по сравнению с исходными требованиями	num
t09	Цели качества программного обеспечения: 1 = очень низкий; нет требований к качеству; 2 = низкий; удовлетворены основные требования (документация, тестирование реализации, тестирование системы и тестирование модулей); нет статистического контроля или обзоров 3 = надлежащая документация по критическим функциям; дизайн и реализация проверены; модули / рабочие потоки протестированы; прохождения; 4 = высокий; формальные обзоры и проверки между всеми этапами; внимание к документации, удобству использования и обслуживанию 5 = очень высокий; количественные требования к качеству; 100% удовлетворение технических и функциональных целей; минимальные работы по техническому обслуживанию	num

Продолжение таблицы 8

Название атрибута	**Описание**	**Тип данных**
t010	Цели эффективности программного обеспечения: 1 = очень низкий; нет требований к эффективности, требующих внимания или планирования 2 = Низкий; легко достижимые цели эффективности; требования ниже среднего 3 = номинальный; уровень мощности программного обеспечения стабильный и предсказуемый; время отклика, нагрузка транзакции и время выполнения работ 4 = высокий; конкретные пики мощности, времени отклика, достигаемые с помощью конкретных методов проектирования и реализации 5 = очень высокий; строгие цели эффективности, требующие постоянного внимания и особых навыков	num
t11	Потребности в обучении пользователей и варианты платформы: 1 = очень низкий; нет необходимости в обучении; <10 пользователей 2 = Низкий; некоторое обучение; около 10 пользователей; создание основных данных только второстепенное 3 = номинальный; типичное обучение; 10-50 пользователей; некоторые преобразования старых данных 4 = высокий; масштабное обучение для нескольких организации; <1000 пользователей; дополнительное программное обеспечение для конвертации; возможные параллельные прогоны; несколько платформ 5 = очень высокий; > 1000 пользователей; ожидаемый долгий срок службы; несколько пользовательских организаций; несколько разных платформ	num

Продолжение таблицы 8

Название атрибута	**Описание**	**Тип данных**
t12	Аналитические навыки сотрудников проекта на старте: 1 = очень низкий; нет опыта в анализе требований или подобных проектах 2 = Низкий; <30% проектного персонала с опытом анализа и проектирования в аналогичных проектах 3 = номинальный; 30-70% сотрудников проекта с опытом анализа; один опытный участник 4 = высокий; большинство сотрудников, имеющих опыт работы со спецификациями и анализом; ответственный специалист по анализу 5 = очень высокий; штат проекта, состоящий из первоклассных профессионалов; у участников есть четкое видение и опыт анализа требований	num
t13	Знание предметной области в проектной команде (поставщик и заказчик): 1 = очень низкий; опыт работы в команде <6 месяцев в среднем 2 = Низкий; опыт применения низкий; у некоторых участников есть опыт; 6-12 месяцев в среднем 3 = номинальный; опыт применения хороший; 1-3 года в среднем 4 = высокий; хороший опыт применения как у поставщиков, так и у клиентов; В среднем 3-6 лет; известна динамика бизнеса 5 = очень высокий; и поставщик, и заказчик хорошо знают область применения, в том числе свой бизнес; Средний стаж> 6 лет	num

Продолжение таблицы 8

Название атрибута	**Описание**	**Тип данных**
t14	Уровень опыта проектной команды (поставщик и заказчик) с инструментами разработки и документирования на старте проекта: 1 = очень низкий; команда не имеет опыта работы с необходимыми инструментами; средний опыт команды <6 месяцев 2 = Низкий; инструменты опыт работы меньше среднего; некоторые участники имеют опыт работы с некоторыми инструментами; 6-12 месяцев в среднем 3 = номинальный; инструменты хорошо знакомы примерно с половиной команды; некоторые участники хорошо знают инструменты разработки и документации; 1-3 года в среднем 4 = высокий; большинство членов команды хорошо знают инструменты; некоторые участники могут помогать другим; 3-6 лет в среднем 5 = очень высокий; команда хорошо знает все инструменты; поддерживать	num
t15	Способность проектной команды эффективно работать в соответствии с лучшими проектными практиками: 1 = очень низкий; разрозненная команда; минимальные проектные и управленческие навыки 2 = Низкий; некоторые участники с предыдущим опытом работы в аналогичных проектах; не объединены как группа 3 = номинальный; большинство участников с опытом работы в аналогичных проектах; нет мотивации работы в команде 4 = высокий; группа очень активна и знает, как использовать эффективность команды 5 = очень высокий; опережающая команда; команда может инновационным образом решать личные и командные задачи;	num

3.3.3 Предобработка данных

Перед применением методов машинного обучения к выбранным датасетам для получения более качественных результатов оценки трудоемкости разработки программного обеспечения корпоративных информационных систем нужно выполнить подготовить данные.

Подготовка данных - это процесс преобразования необработанных данных для получения прогнозов. Она дает чистые и тщательно отобранные данные, которые приводят к более практичным и точным результатам модели. Процесс подготовки записей датасетов для применения к выбранным методам состоял из следующих этапов:

1. отсутствующие записи строкового типа были заменены на наиболее распространенное значение в каждом столбце,
2. все строковые записи были преобразованы в числовые, которые наши предсказательные модели понимают лучше,
3. были удалены столбцы, которые очевидно не влияют на результат прогнозирования, например, идентификатор записи,
4. были удалены атрибуты, которые имели наименьшую корреляцию со значением оценки трудоемкости.

Все этапы предобработки датасетов проводились на языке Python с помощью библиотеки scikit-learn.

На рис. 9, 10, 11 представлены корреляционные матрицы для каждого из рассматриваемых датасетов. Для датасета Maxwell были оставлены атрибуты, имеющие значение корреляции с атрибутом effort больше 0,3. Для датасета SiP были оставлены атрибуты, имеющие значение корреляции с атрибутом HoursActual больше 0,05 по причине низкой корреляции признаков. Для датасета Desharnais были оставлены атрибуты, имеющие значение корреляции с атрибутом Effort больше 0,4.

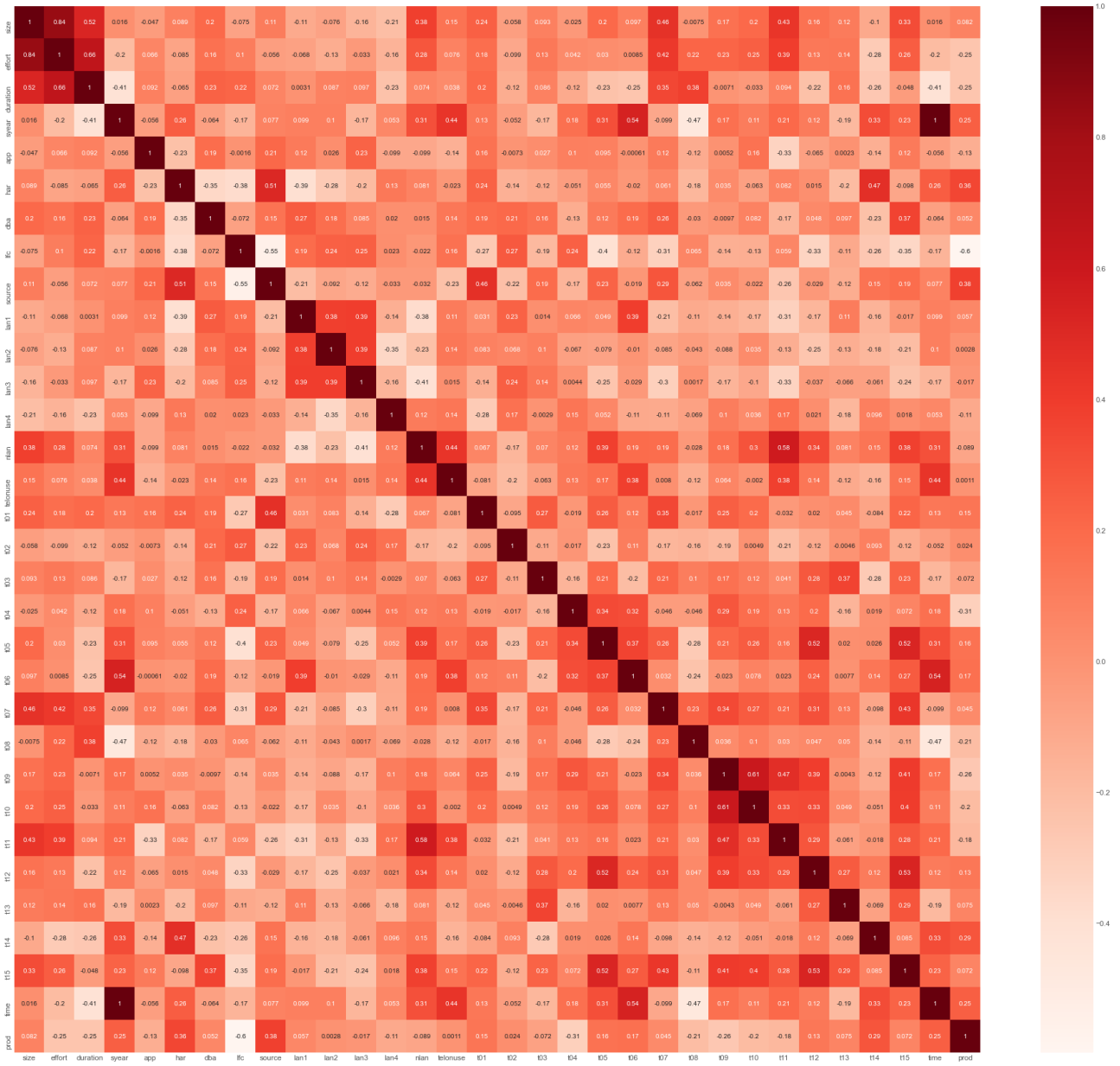

Рис. 9. Матрица корреляции датасета Maxwell

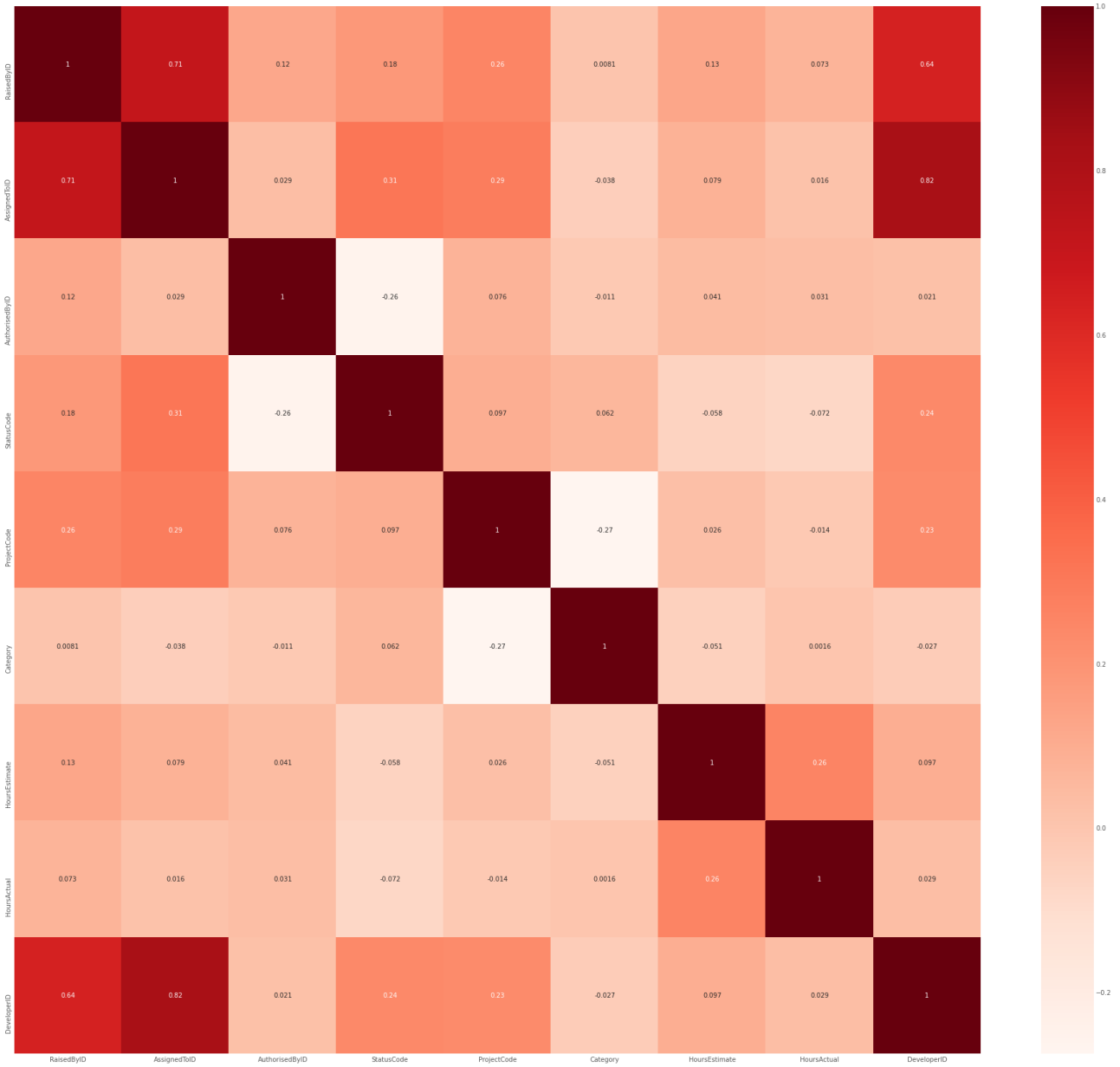

Рис. 10. Матрица корреляции датасета SiP

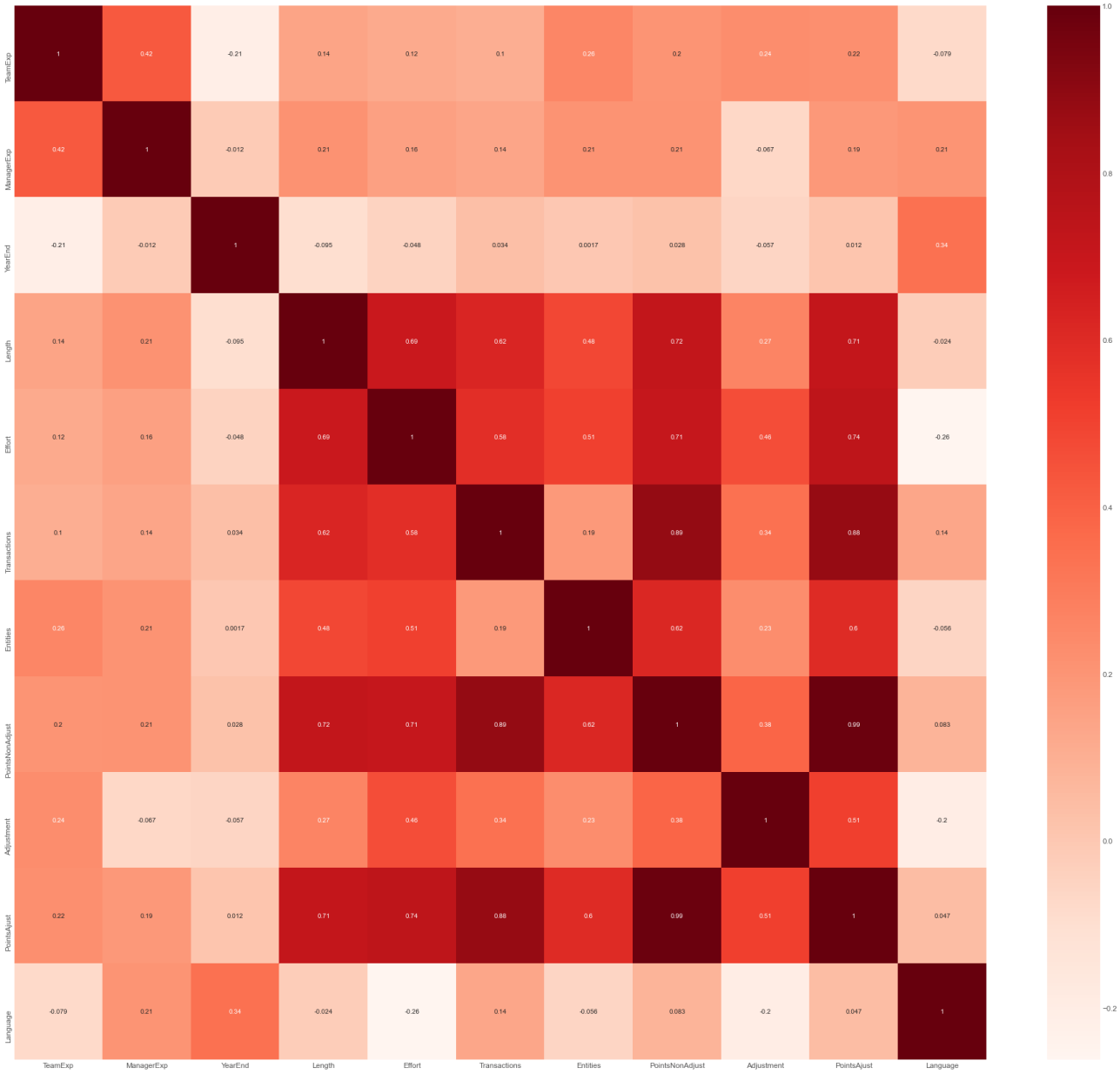

Рис. 11. Матрица корреляции датасета Desharnais

3.4 Описание параметров модели

Для данного эксперимента использовались следующие параметры моделей машинного обучения:

– регрессия опорных векторов (SVR) с радиальной базисной функцией (RBF) в качестве ядра и коэффициентом ядра gamma = 0.1,

– случайный лес (RF) со 100 деревьями и случайностью начальной загрузки образцов, используемых при построении деревьев равным 42.

– Обобщенная регрессионная нейронная сеть (GRNN) с радиальной базисной функцией (RBF) в качестве ядра,
– линейная нейронная сеть (NN) с входным слоем, двумя скрытыми слоями и одним выходным. Количество ядер в каждом слое зависело от датасета: Desharnais: входной слой – 7 ядер, второй слой – 7 ядер, третий слой – 1000 ядер, выходной слой – 1 ядро; SiP Dataset: входной слой – 8 ядер, второй слой – 8 ядер, третий слой – 10000 ядер, выходной слой – 1 ядро; Maxwell: входной слой – 11 ядер, второй слой 11 ядер, третий слой – 1000 ядер, выходной слой – 1 ядро,
– сеть радиально-базисных функций (RBFN) с входным слоем, одним скрытым слоем и одним выходным,
– в ансамбле (Ensemble) в качестве базовых алгоритмов использовались: случайный лес со 100 деревьями и случайностью начальной загрузки образцов, используемых при построении деревьев равным 42; регрессия опорных векторов Регрессия опорных векторов (SVR) с радиальной базисной функцией (RBF) в качестве ядра и коэффициентом ядра gamma = 1/(количество атрибутов*дисперсия входного массива); линейная нейронная сеть: входной слой зависит от количества атрибутов датасета, второй слой – 512 ядер, третий слой – 256 ядер, четвертый слой – 64 ядра, выходной слой – 1 ядро.

В качестве мета алгоритма по умолчанию в библиотеке scikit-learn используется ридж-регрессия (ridge regression).

Код анализатора для рассматриваемых датасетов доступен в репозитории на Github [33].

3.5 Реализация системы

3.5.1 Средства реализации

Python используется в качестве сервера, в него будут поступать данные для анализа и он будет взаимодействовать с СУБД. Для разработки веб-приложений на Python используется микро-веб-фреймворк Flask. Он масштабируем и прост в использовании по сравнению со своими аналогами.

Для внешнего интерфейса используется Angular, один из самых популярных фреймворков. Он предоставляет разработчикам инструменты, необходимые для создания и структурирования крупномасштабных приложений JavaScript. Кроме того, у Angular есть большие преимущества перед некоторыми альтернативами. Например, он создается и поддерживается инженерами Google.

В качестве СУБД используется PostgreSQL.

3.5.2 Реализация доступа к данным

Для взаимодействия базы данных и Python была применена библиотека SQLAlchemy. Она используется в качестве инструмента объектно-реляционного сопоставления (ORM), который переводит классы Python в таблицы в реляционных базах данных и автоматически преобразует вызовы функций в операторы SQL. Преимущество данной библиотеки в универсальности работы с различными базами данных. Взаимодействие с разными ядрами СУБД происходит через один и тот же API.

Для таблиц User, Project и AnalyzationResult были созданы следующие классы:

```
class User(db.Model):
  __tablename__ = 'user'
  id = db.Column(db.Integer, primary_key=True)
  login = db.Column(db.String, unique=True)
  password = db.Column(db.String)
```

```
  project = db.relationship('Project')


class Project(db.Model):
  __tablename__ = 'project'
  id = db.Column(db.Integer, primary_key=True)
  user_id = db.Column(db.Integer, db.ForeignKey('user.id'))
  data_source = db.Column(db.String)


class AnalyzationResult(db.Model):
  __tablename__ = 'analyzation-result'
  id = db.Column(db.Integer, primary_key=True)
  project_id = db.Column(db.Integer, db.ForeignKey('project.id'))
  result = db.Column(db.String)
```

Для подключения к PostgreSQL используется следующая строка:

```
'postgresql://{db_user}:{db_password}@{db_url}/{db_name}'
```

где

– db_user – имя пользователя,
– db_password – пароль пользователя для получения доступа к базе данных,
– db_url – URL для порта, в котором запущена база данных,
– db_name – название базы данных, доступ к которой необходимо получить.

3.5.3 Авторизация в системе

Для просмотра существующих прогнозов и получению новых пользователю необходимо пройти процедуру аутентификации в системе. Чтобы сервер мог определить, исходит ли запрос от авторизованного пользователя, добавляется дополнительный HTTP-заголовок - JWT токен.

JSON Web Token (JWT) - это открытый стандарт (RFC 7519), определяющий компактный и автономный способ безопасной передачи информации между сторонами в виде объекта JSON. После входа пользователя в систему каждый последующий запрос будет включать JWT, позволяя пользователю получать доступ к маршрутам, службам и ресурсам, разрешенным с помощью этого токена.

Веб-токены JSON состоят из трех частей, разделенных точками:

1. Заголовок, состоящий из типа токена, которым является JWT, и используемого алгоритма подписи
2. Полезная нагрузка, в нее помещаются данные, которые необходимо безопасно передать: имя пользователя, метка времени выдачи и метка времени истечения срока действия
3. Подпись, сгенерированная с помощью алгоритмов HMAC и SHA-256. Подпись гарантирует, что токен был создан известной стороной и целостность токена

Для отправки данных на сервер с токеном необходимо перехватывать HTTP-запросы перед их отправкой и вносить в них необходимые изменения. Для этого создан это HttpInterceptor с методом перехвата. В нем проверяется доступность токена и его установка в соответствующий HTTP-заголовок.

```
@Injectable()
export class TokenInterceptor implements HttpInterceptor {
 constructor(public authService: AuthService) { }
```

```
    intercept(request: HttpRequest<any>, next: HttpHandler):
Observable<HttpEvent<any>> {

      if (this.authService.getJwtToken()) {

        request = this.addToken(request, this.authService.getJwtToken());

      }

      return next.handle(request).pipe(catchError(error => {

        if (error instanceof HttpErrorResponse && error.status === 401) {

          return this.handle401Error(request, next);

        } else {

          return throwError(error);

        }

      }));

    }
```

При поступлении входящего запроса с токеном доступа, который стал недействительным, приложение может отправить токен обновления для получения нового токена доступа. Если сеанс пользователя все еще активен, сервер ответит новым действующим JWT.

Внешний вид страницы входа представлен на рис. 12.

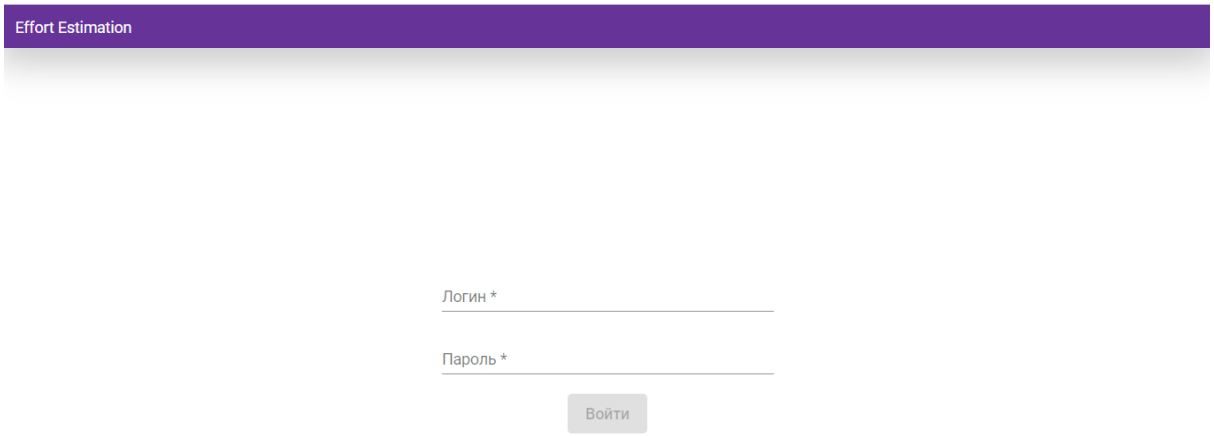

Рис. 12. Страница входа в веб-систему

3.5.4 Реализация анализатора

После логина отображается главная страница, представленная на рис. 3.11.

Для предоставления Python как REST сервера используется пакет Flask. В таблице 9 представлены запросы, интерпретируемые анализатором.

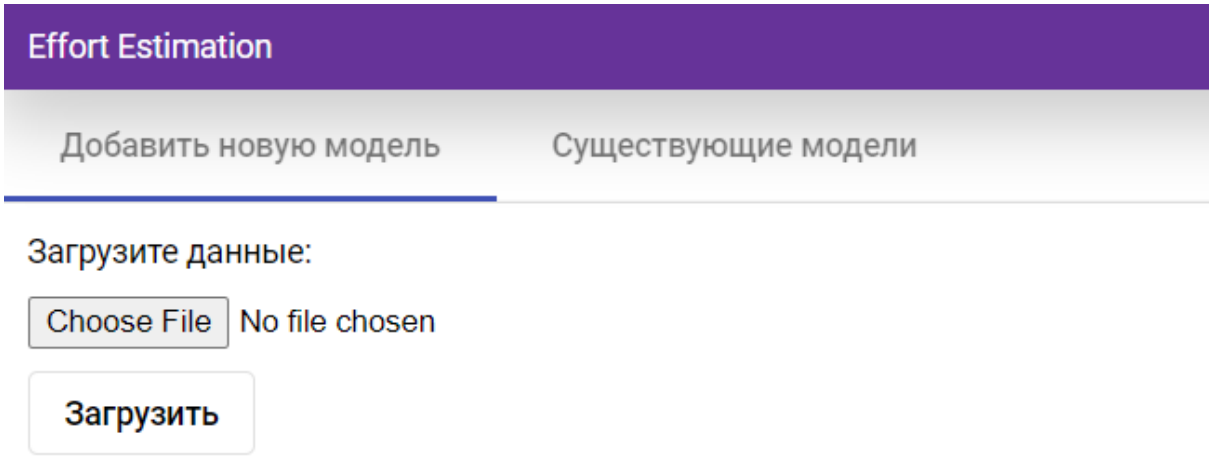

Рис. 13. Главная страница веб-системы

Таблица 9 - REST запросы

Тип	**URI**	**Описание**
GET	/models	получение моделей
GET	/models/{id}	получение модели
POST	/models	создание модели
DELETE	/models/{id}	удаление модели
POST	/models/{id}/analyzation-result	получение прогноза

Для подбора оптимальных параметров построения модели лучшие параметры выбираются автоматически с помощью кросс-валидации, предоставляемой библиотекой для Python scikit-learn.

При получении запроса на создание модели анализатор обучается на предложенном датасете и выдает результат работы. Пример результата работы представлен на рис. 3.14.

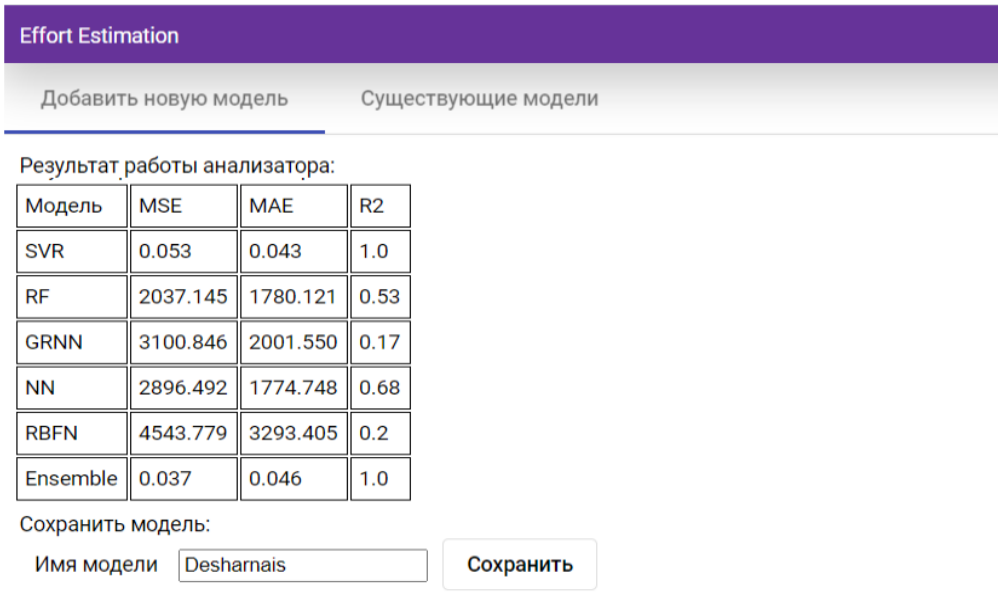

Рис. 14. Отображение результата работы анализатора

Для получения прогноза существующих моделей пользователь выбирает проанализированный датасет и модель, которая будет рассчитывать

прогноз. Далее пользователь загружает файл формата .csv, в котором записаны параметры для построения прогноза. В результате работы выводится спрогнозированное значение. Интерфейс для работы с существующими моделями представлен на рис. 3.15.

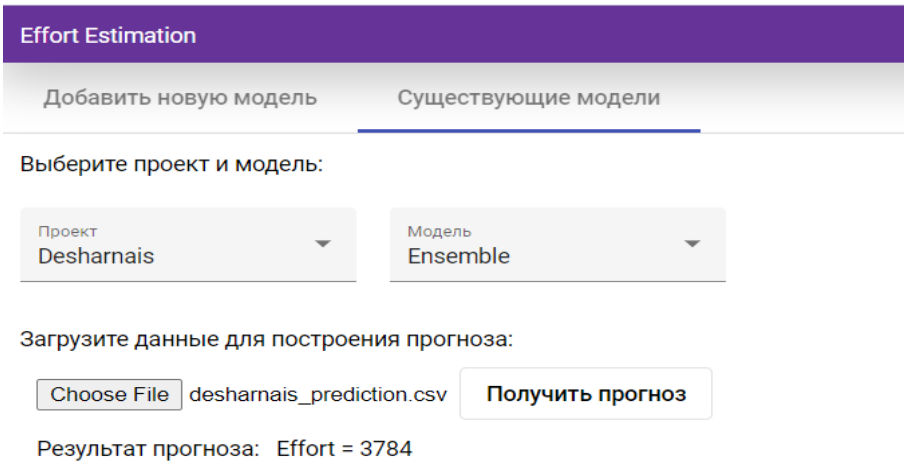

Рис. 15. Интерфейс для получения прогноза

3.6 Вывод

В данной главе были представлены требования к разрабатываемой системе для оценки трудоемкости разработки программного обеспечения корпоративных информационных систем с использованием методов машинного обучения. Также были представлены компоненты системы, схема базы данных. Был проведен эксперимент над тремя датасетами с применением выбранных моделей машинного обучения на языке Python. Была разработана веб система для получения оценки трудоемкости.

Глава 4 Анализ результатов исследования

4.1 Сравнение моделей машинного обучения

Было проведено исследование методов машинного обучения в оценке трудоемкости программного обеспечения корпоративных информационных систем. Обзор статей показал, что чаще всего для исследования проблемы применяются регрессионные модели и нейронные сети. Чаще всего из ансамблевых методов применяется бэггинг. Ансамбль стекинг с применением алгоритма случайного леса, нейронной сети и регрессии опорных векторов нигде не применялся.

Для сравнения вышеописанных моделей решения задачи трудоемкости разработки программного обеспечения мною были выбраны три критерия: устойчивость к шуму во входных данных, вычислительная сложность алгоритмов, требования к количеству обучающих примеров, необходимых для качественного обучения.

На реальные данные, которые являются входными для алгоритмов интеллектуального анализа данных, влияют несколько компонентов; среди них присутствие шума является ключевым фактором [31]. Шум - это все, что является ложным и посторонним по отношению к исходным данным, которое изначально не предназначено для присутствия, но было внесено из-за неправильного процесса получения данных. Это неизбежная проблема, которая влияет на процессы сбора и подготовки данных в приложениях анализа данных, где часто возникают ошибки. У шума два основных источника [32]: ошибки ввода измерительными инструментами, например, различными типами датчиков; и случайные ошибки, вносимые пакетными процессами или экспертами при сборе данных.

Вычислительная сложность алгоритмов машинного обучения - это функция зависимости объема работы, которая выполняется алгоритмом, от размера входных данных. Для данной оценки использовалась О-нотация. В

этой нотации используется специальная математическая функция от n, т.е. количества элементов, которой пропорционально быстродействие алгоритма.

Данные сравнения моделей представлены в таблице 10.

Таблица 10 - Сравнение рассматриваемых моделей машинного обучения

Модель МО	**Устойчивость к шуму во входных данных**	**Вычислительные затраты**	**Требования к количеству обучающих примеров**
SVR	неустойчив	$O(n_{sv}p)$ n_{sv} - количество опорных векторов, p – количество входных атрибутов	хуже работает с большим количеством
RF	склонен к переобучению	$O(pn_{trees})$ n_{trees}- количество деревьев, p – количество входных атрибутов	эффективен с большим количеством
ANN	устойчив	$O(pn_{l1} + n_{l1}n_{l2} + \cdots)$ n_{li} - количество нейронов в слое i в нейронной сети, p – количество входных атрибутов	требует большого количества
RBFN	устойчив		требует большого количества
GRNN	устойчив		требует меньшего количества

Продолжение таблицы 10

Модель МО	**Устойчивость к шуму во входных данных**	**Вычислительные затраты**	**Требования к количеству обучающих примеров**
Stacking ensemble	устойчив	$\max(O_{base-leaners}) + O_{meta-leaner}$ $O_{base-leaners}$ – вычислительные затраты базового алгоритма $O_{meta-leaner}$ – вычислительные затраты мета-алгоритма	лучше работает с большим количеством

4.2 Анализ результатов работы анализатора и перспектив реализованной веб-системы

В рамках диссертации был проведен эксперимент над тремя датасетами. Итоговые результаты работы моделей представлены в таблице 11.

Проведем сравнение полученных результатов с результатами работ других авторов. В работе [14] для датасета Desharnais лучший результат показал метод Support Vector Regression (MAE = 1888.018, RMSE=2361.356, R2 =-0.02994683). Для датасета Maxwell наилучшую работу показал метод ElasticNet Regression (MAE=3113.22, RMSE=4536.323, R2=0.7324483). В работе [12] для датасета Desharnais наилучшим стал метод Linear Regression, как до исключения атрибутов, так и после. Показатель MAE для Linear Regression равен 2013.79, показатель MSE равен 2824.57.

Таблица 11 - Результаты работы моделей с выбранными датасетами

Название датасета	**Среднее значение атрибута в тестовой выборке**	**Метод**	**MSE**	**MAE**	**R2**
Desharnais	4760.0	SVR	0.153	0.103	1.0
		RF	2037.145	1780.121	0.53
		GRNN	3100.846	2001.550	0.17
		NN	2896.492	1774.748	0.68
		RBFN	4543.779	3293.405	0.2
		Ensemble	**0.037**	**0.45**	**1.0**
Maxwell	7757.125	SVR	3338.656	2333.02	0.79
		RF	2315.018	1745.410	0.71
		GRNN	4159.747	3773.190	0.11
		NN	2185.163	1659.027	0.75
		RBFN	3536.700	2776.595	0.58
		Ensemble	**0.012**	**0.017**	**1.0**

Продолжение таблицы 11

Название датасета	**Среднее значение атрибута в тестовой выборке**	**Метод**	**MSE**	**MAE**	**R2**
SiP Dataset	11.765	SVR	54.035	13.473	0.09
		RF	30.374	4.450	0.56
		GRNN	41.553	14.301	0.3
		NN	50.436	16.802	0.2
		RBFN	38.694	12.527	0.4
		Ensemble	**1.045**	**4.639**	**1**

Проведем сравнение полученных результатов с результатами работ других авторов. В работе [14] для датасета Desharnais лучший результат показал метод Support Vector Regression (MAE = 1888.018, RMSE=2361.356, R2 =-0.02994683). Для датасета Maxwell наилучшую работу показал метод ElasticNet Regression (MAE=3113.22, RMSE=4536.323, R2=0.7324483). В работе [12] для датасета Desharnais наилучшим стал метод Linear Regression, как до исключения атрибутов, так и после. Показатель MAE для Linear Regression равен 2013.79, показатель MSE равен 2824.57.

Таким образом в рамках эксперимента с датасетами было показано, что предложенный ансамблевый метод показал лучшие результаты для Desharnais и Maxwell датасетов.

Алгоритм показал наиболее эффективный результат для датасетов,у которых были большие корреляции с пронзируемым знаечнием. В SiP датасете результаты хуже (в процентном соотношении), поскольку поазатедли корреляции ниже со всеми аттрибутами (ниже 0.26).

В будущем для более точного прогнозирования при составлении исторических записей нужно учитывать больше данных, имеющих высокую корреляцию с показателем трудоемкости.

Была разработана веб-система, в которых применялись разработанные модели.

4.3 Рекомендации по практическому применению полученных результатов и перспектива развития исследования

Результаты диссертации показывают, что такой инструмент как машинное обучение стоит применять в оценке трудоемкости разработки корпоративных информационных систем. Мной была разработана ансамблевая модель, которая показывает высокую эффективность ее использования при прогнозировании оценки трудоемкости разработки корпоративных информационных систем.

В качестве дальнейших исследований можно рассмотреть улучшение прогнозирования. Так же применение моделей не только для оценки трудоемкости проекта, но и задач внутри проекта. А также внедрение анализатора в существующие корпоративные информационные системы.

Таким образом, в качестве перспективы развития веб-системы можно рассматривать получение прогноза с различающимися атрибутами.

Заключение

В рамках исследования был проведен анализ литературы и электронных источников, касающихся использования методов машинного обучения в оценке трудоемкости разработки. По итогам статистических данных был сделан вывод, что проблема оценки трудоемкости программного обеспечения является актуальной темой. Применение методов машинного обучения для получения оценки трудоемкости разработки корпоративных информационных систем дает следующие преимущества: возможность обрабатывать большие наборы данных, возможность вовремя заметить и проанализировать изменения в процессе, возможность объективно и точно оценить результат, возможность свести к минимуму ошибки, возникающих в результате человеческого фактора.

Был проведен обзор методов машинного обучения для оценки трудоемкости разработки корпоративных систем. В процессе анализа научных статей был сделан вывод о том, что чаще всего для оценки трудоемкости используются классические методы машинного обучения, ансамблевые методы используются реже, поскольку они сложны в проектировании, долго обучаются.

Был проведен обзор существующих корпоративных систем. Имеющиеся системы не применяют методы машинного обучения для получения оценки трудоемкости разработки, поскольку это достаточно новая технология.

Был проведен предварительный эксперимент на данных с использованием выбранных методов машинного обучения. В рамках эксперимента был определен оптимальный метод для прогнозирования трудоемкости с использованием классических методов машинного обучения, нейронных сетей и ансамбля методов, который позволяет обеспечить точность прогнозирования трудоемкости с наименьшим значением средней квадратичной ошибки (MSE). Результаты эксперимента показали, что

предложенный ансамблевый метод значительно превосходит по качеству другие рассматриваемые методы машинного обучения. Для датасета Desharnais ансамбль лучше по сравнению с лучшим методом (SVR), помимо ансамбля, в 4,135 раз по средней квадратичной ошибке. Для Maxwell датасета ансамбль лучше по сравнению с лучшим методом (NN), помимо ансамбля, в 182096,92 раз по средней квадратичной ошибке. Для датасета SiP ансамбль лучше по сравнению с лучшим методом (RF), помимо ансамбля, в 29,06 раз по средней квадратичной ошибке.

Была спроектирована и реализована веб-система с применением созданных модели.

Для достижения поставленной цели исследования были решены все поставленные задачи. Таким образом, можно утверждать, что цель исследования была достигнута.

Список используемых источников

1. Yang, DA. Et al (2008) “A Survey on Software Cost Estimation in the Chinese Software Industry”. Of: ESEM -08: Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, pp.253-262.
2. 5 Worldwide Semiannual Digital Transformation Spending Guide, International Data Corp., 2018
3. Pulse of the Profession 2016 (2016), PMI
4. Pulse of the Profession 2016, PMI
5. Pulse of the Profession, 2018, PMI
6. Pulse of the Profession, 2019, PMI
7. Banimustafa, Ahmed. (2018). Predicting Software Effort Estimation Using Machine Learning Techniques. 249-256.
8. Braga, Petrônio & Oliveira, Adriano & Meira, Silvio. (2007). Software Effort Estimation using Machine Learning Techniques with Robust Confidence Intervals. 352-357.
9. Alhazmi, Omar & Khan, Mohammed. (2020). Software Effort Prediction Using Ensemble Learning Methods. Journal of Software Engineering and Applications. 13. 143-160.
10. Carvalho, Halcyon & Lima, Marília & Santos, Wylliams & Fagundes, Roberta. (2020). Ensemble Regression Models for Software Development Effort Estimation: A Comparative Study.
11. Singh, Mukesh & Kumar, Mukesh. (2020). Comparative Study On Effort Estimation Using Different Data Mining Techniques. International Journal of Scientific & Technology Research.
12. Singh, A. & Kumar, Mukesh. (2020). Comparative Analysis on Prediction of Software Effort Estimation Using Machine Learning Techniques. SSRN Electronic Journal.

13. Nassif, Ali & Azzeh, Mohammad & Capretz, Luiz & Ho, Danny. (2016). Neural network models for software development effort estimation: a comparative study. Neural Computing and Applications. 27.
14. Varshini, A. (2020). Predictive analytics approaches for software effort estimation: A review. Indian Journal of Science and Technology. 13. 2094-2103.
15. GoodDay Platform [Электронный ресурс] // URL: https://www.goodday.work/ (дата обращения: 20.10.2019)
16. Jira Software [Электронный ресурс] // URL: https://www.atlassian.com/ru/software/jira (дата обращения: 20.10.2019)
17. COCOMO II Web Tool [Электронный ресурс] // URL: http://softwarecost.org/tools/COCOMO/ (дата обращения: 20.10.2019)
18. SLIM-Estimate Tool [Электронный ресурс] // URL: https://www.qsm.com/tools/slim-estimate (дата обращения: 20.10.2019)
19. Drucker, Harris; Burges, Christ. C.; Kaufman, Linda; Smola, Alexander J.; and Vapnik, Vladimir N. (1997); "Support Vector Regression Machines", in Advances in Neural Information Processing Systems 9, NIPS 1996, 155–161, MIT Press.
20. Leo Breiman. Random Forests // Machine Learning, October 2001, Volume 45, Issue 1, pp 5-32.
21. Breiman L. Bagging predictors // Machine Learning, 1996, vol. 24, no. 2, pp. 123–140.
22. Caruana R., Niculescu-Mizil A. An Empirical Comparison of Supervised Learning Algorithms // Department of Computer Science, Cornell University, Ithaca, NY 14853 USA.
23. Mark R. Segal. Machine Learning Benchmarks and Random Forest Regression // Division of Biostatistics, University of California, San Francisco, CA 94143-0560, April 14, 2003.

24. M.J.D. Powell, ”Radial basis functions for multivariable interpolation: A review”, in IMA Conf. on Algorithms for the approximation of functions and data, pp. 143-167, 1985.
25. D. F. Specht, “A General Regression Neural Network”, Neural Networks, IEEE Transactions on, vol. 2, 1991, pp. 568-76.
26. S. E. Fahlman and C. Lebiere, The cascadecorrelation learning architecture, Technical Report CMU-CS-90-100, School of Computer Science, Carnegie Mellon University, 1991.
27. M. Hoefeld and S. E. Fahlman, Learning with limited numerical precision using the cascadecorrelation algorithm, Technical Report CMUCS-91-130, School of Computer Science, Carnegie Mellon University, 1991.
28. S. E. Fahlman, The recurrent cascadecorrelation architecture, Technical Report CMU-CS-91-100, School of Computer Science, Carnegie Mellon University, 1991.
29. T. R. Schultz, F. Rivest, L. o Egri, J. P. Thivierge, Knowledge-based Learning with KBCC, Proceedings in the 5th International Conference on Development and Learning, 2006.
30. RY Wang, VC Storey, CP Firth, A Framework for Analysis of Data Quality Research, IEEE Transactions on Knowledge and Data Engineering 7 (1995) 623-640.
31. Wolpert, David. (1992). Stacked Generalization. Neural Networks. 5. 241-259
32. Sayyad Shirabad, J. and Menzies, T.J. (2005) The PROMISE Repository of Software Engineering Databases. School of Information Technology and Engineering, University of Ottawa, Canada Available: http://promise.site.uottawa.ca/SERepository
33. Github репозиторий [Электронный ресурс] // URL: https://github.com/KhMaria/effort-esimation-ml (дата обращения: 01.04.2021).