МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра вычислительных технологий**

**КУРСОВАЯ РАБОТА**

**РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ ОЦЕНКИ КАДАСТРОВОЙ СТОИМОСТИ ЖИЛЬЯ В КРАСНОДАРЕ**

Работу выполнила \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.С. Агаджанян

(подпись)

Направление подготовки 02.03.02 — «Фундаментальная информатика и\_\_\_\_\_

(код, наименование)

информационные технологии»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_курс\_\_\_\_3\_\_\_\_\_

Направленность (профиль) \_\_\_\_\_\_\_\_\_\_\_\_Математическое и программное обеспечение компьютерных технологий\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Научный руководитель

канд. техн. наук, доц. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Т. А. Приходько

(подпись, дата)

Нормоконтролер

преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Е. А. Нигодин

(подпись, дата)

Краснодар

2025

**РЕФЕРАТ**

Курсовая работа 26 стр., 4 ч., 9 рис., 26 источ., 1 приложение.

Курсовая работа посвящена разработке и оценке модели для автоматизированной оценки кадастровой стоимости жилья. Актуальность работы обусловлена необходимостью повышения точности и объективности кадастровой оценки, которая влияет на налогообложение и управление недвижимостью.

Целью работы является разработка и исследование эффективности модели оценки кадастровой стоимости жилья на основе выбранного метода. Для достижения цели были решены следующие задачи:

* анализ существующих подходов к кадастровой оценке
* выбор оптимального метода
* разработка программной реализации модели
* экспериментальная оценка точности модели на реальных данных и анализ полученных результатов.

**Объект исследования:** процесс оценки кадастровой стоимости жилья с использованием информационных технологий. Более конкретно - модели и методы, применяемые для автоматизированной оценки кадастровой стоимости жилой недвижимости.

**Предмет исследования:** метод RandomForestRegressor и его применимость для построения модели оценки кадастровой стоимости жилья, а также факторы, влияющие на точность оценки с использованием данного метода

**СОДЕРЖАНИЕ**

[Введение](#_Toc121868742) 4

[1  Теоретические основы оценки кадастровой стоимости жилья](#_Toc121868743) 5

[1.1 Основные понятия машинного обучения для оценки недвижимости](#_Toc121868744) 5

[1.2 Алгоритм RandomForestRegressor: принцип работы и особенности](#_Toc121868746) 11

[1.3 Коэффициент детерминации……………………………………………….13](#_Toc121868746)

[2 Архитектура и реализация веб-приложения](#_Toc121868749) 15

[2.1 Обзор архитектуры Model-View-Template (MVT)](#_Toc121868750) 15

[3 Программная реализация](#_Toc121868752) 17

[3.3 Описание функциональности приложения](#_Toc121868755) 17

[4 Исследовательская часть](#_Toc121868756) 19

[4.1 Результаты работы приложения](#_Toc121868757) 19

[Заключение](#_Toc121868761) 23

[Список использованных источников](#_Toc121868762) 24

Приложение A……………………………………………………………………27

**ВВЕДЕНИЕ**

Актуальность темы курсовой работы обусловлена возрастающей потребностью в точной и объективной оценке кадастровой стоимости жилья. Кадастровая стоимость является базой для расчета налогов на недвижимость, арендной платы, а также используется при определении стоимости при купле-продаже и других сделках. Недостаточная точность кадастровой оценки может приводить к несправедливому налогообложению, судебным разбирательствам и снижению инвестиционной привлекательности. Традиционные методы оценки, основанные на экспертных оценках и устаревших подходах, часто не учитывают все факторы, влияющие на стоимость, и могут быть подвержены субъективности.

В последние годы наблюдается активное внедрение технологий машинного обучения (Machine Learning, ML) в различных областях, включая оценку недвижимости. Методы ML, такие как регрессионные модели, позволяют учитывать широкий спектр факторов, влияющих на стоимость, и обеспечивают более высокую точность оценки по сравнению с традиционными подходами.

Целью данной курсовой работы является разработка и исследование модели оценки кадастровой стоимости жилья на основе метода случайного леса (RandomForestRegressor).

Объектом исследования является процесс оценки кадастровой стоимости жилья.

Предметом исследования является метод RandomForestRegressor и его применимость для построения модели оценки кадастровой стоимости жилья, а также факторы, влияющие на точность оценки жилья, которая может быть использована для повышения точности и объективности оценки, оптимизации налогообложения и принятия обоснованных решений в сфере управления недвижимостью.

**1 Теоретические основы оценки кадастровой стоимости жилья**

* 1. **Основные понятия машинного обучения для оценки недвижимости**

Для более глубокого понимания прогнозирования стоимости недвижимости проведен анализ существующих исследований и научных работ в данной области. Литературный обзор позволил выявить основные подходы к анализу рынка недвижимости, методы прогнозирования цен на жилье, а также принципы построения моделей машинного обучения. Исследование информации проводилось с помощью общедоступных источников в сети интернет с базами данных научных публикаций Scopus, Scholar, Cyberleninka, Elibrary. Поиск проводился по ключевым словам: «исследование индекса цен на квартиры на основе машинного обучения», «прогнозирование цен недвижимости», «living houses cost prediction», «критерии оценки моделей машинного обучения недвижимости».

Аналитические методы оценки включают различные техники. Логическая экстраполяция текущих данных для прогнозирования будущих ценовых трендов, известная как эвристический метод, становится всё популярнее. Сравнение продаж помогает определить относительную стоимость похожих объектов в одном регионе. Количественный статистический анализ данных позволяет выявить текущее состояние рынка и отследить изменения во времени. В России часто используется доходнозатратный подход, который учитывает потенциальную прибыль от использования объекта и стоимость его строительства или замены. Эти методы дополняются сравнением продаж и гедонистическим моделированием ценообразования [5]. Существует множество методов машинного обучения, которые теоретически могут использоваться для определения рыночной стоимости недвижимости. В практике оценки часто применяется линейная регрессия, классические алгоритмы машинного обучения, такие как Случайный лес и Gradient boosting, а также более современные модели, включая XGBoost и Catboost, и нейронные сети [6]. В статье Никитиной Н.С. определены преимущества и недостатки каждой из моделей:

1. ANN (Artificial neural network) [7] – имеет низкую точность прогнозирования, позволяет использовать нелинейные функции;
2. HPM (Hedonic price model) [8] – адаптивна к добавлению различных рыночных товаров и учету внешних факторов;
3. FLS (Fuzzy logic system) [9] – альтернативная модель традиционной оценки имущества, снижающая неопределенность прогноза, но не учитывающая факторы риска;
4. SVM (Support vector machine) [10] может помочь найти оптимальное решение, но в ограниченном объеме выборки;
5. LR (Линейная регрессия) [11] требует наличия предпосылки о структуре данных и использует линейную форму функции, которая неизменна во времени;
6. DT (Decision tree) [12] подходит для выбора переменных и работы с категориальными данными, но может приводить к созданию сложного «дерева решений», результаты которого невозможно интерпретировать;
7. RF (Случайный лес) [13] обладает высокой̆ точностью, но набор данных формируется случайно;
8. KNN (K-nearest neighbour) [14] учитывает меру сходства на основе функции расстояния от похожего по исходным характеристикам объекта, чтобы получить наиболее точную оценку стоимости, поэтому возникает проблема поиска таких соответствий;
9. PLS (Partial least square) [15] подходит для работы с коррелированными переменными, но может быть ненадежна, когда данные не распределены нормально;
10. NB (Naive bayes) [16] – простая в использовании модель, не требующая большого количества данных;
11. MRA (Multiple regression analysis) [17] позволяет оценивать коэффициенты с большим количеством данных, но при наличии коррелированных переменных оценка может быть невозможна;
12. SA (Spatial analysis) [18] сложно проводить из-за требуемого качества пространственных данных;
13. GB (Gradient boosting) [19] работает лучше, чем RF (Случайный лес), но требует качественных данных;
14. Ridge и Lasso regressions [20] позволяют работать с мультиколлинеарностью, но со смещённой оценкой дисперсии;
15. ELM (Ensemble learning model) [21] обладает высокой точностью и надежностью прогнозирования, но чувствительна к наличию наблюдений, существенно отличающихся по характеристикам от общей выборки.

Далее, в таблице 1, представлен сравнительный анализ методов прогнозирования на основании изученных публикаций (для некоторых публикаций приведена часть показателей как пример):

Таблица 1 – Обзор публикаций

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Статья, год | Данные, объем | Результаты метрик | Тип алгоритма машинного обучения | Комментарий |
| [2], 2022 | Датасет по аренде офисов за период 2015- 2021 в КуалаЛумпур, Малайзия | R 2 = 0,852, RMSE = 80,663 | Decision Tree | Алгоритм Случайный лес превзошел другие алгоритмы в отношении прогноза стоимости аренды офисов. Он выдал наиболее точный прогноз по сравнению с Decision Tree и Support Vector Machine |
| R 2 = 0,883, RMSE = 75,695 | Случайный лес |
| R 2 = 0,479, RMSE = 155,465 | Support Vector Machine |
| [5], 2023 | - | - | ARIMA для детектирования исторически индуцированных закономерностей, построения прогноза | Метод анализа временных рядов |
| [6], 2020 | Квартиры г. Нижний Новгород, 9 050 ед. | R 2 = 0,66, MAPE = 11,5, MedAPE = 8,9 | Линейная регрессия | Проведенные исследования позволили получить точность 5,8 (MedAPE) на тестовой выборке с помощью алгоритма Случайный лес |
| R 2 = 0,81, MAPE = 8,1, MedAPE = 5,8 | Случайный лес |
| R 2 = 0,78, MAPE = 8,6, MedAPE = 6,4 | Gradient Boosting |
| R 2 = 0,78, MAPE = 9,0, MedAPE = 6,9 | XGBoost |
| R 2 = 0,77, MAPE = 9,1, MedAPE = 7,1 | CatBoost |
| R 2 = 0,71, MAPE = 9,4, MedAPE = 7,1 | Neural Network |
| [22], 2023 | Датасет (Москва, СанктПетербург, Республика Крым), 428 488 ед | R 2 = 0,58, MAE = 0,84, MSE = 0,013 | Линейная регрессия | В этой задаче Gradient Boosting может считаться лучшим подходом с точки зрения точности прогнозов, а Линейная регрессия – менее точным подходом |
| R 2 = 0,659, MAE = 0,074, MSE = 0,011 | Decision Tree |
| R 2 = 0,685, MAE = 0,069, MSE = 0,01 | Gradient Boosting |
| R 2 = 0,681, MAE = 0,071, MSE = 0,01 | Случайный лес |
| [23], 2023 | Kaggle.com, Набор данных 5,4 млн. записей | RMSE = 1173733.8482 | XGBoost | В среднем модель XGBoost лучше справилась с обучением, чем LGBM |
| RMSE = 1213719,1148 | LGBM |
| [23], 2017 | Недвижимость г. Екатеринбург, 2 360 ед. | R 2 = 0,87 на тестовой выборке | Нейронной сеть: персептрон с пятнадцатью входными нейронами, одним выходным нейроном и тремя нейронами скрытого слоя | Пригодна для среднесрочного прогнозирования и извлечения полезных знаний из выборки |
| [24], 2023 | Квартиры г. Самара, 1 705 ед. | R 2 = 0,991212, MAE = 182577,08, MSE = 209110839446,9 4 | Случайный лес | Наибольшая точность предсказания у Случайный лес |
| R 2 = 0,990235, MAE = 213013,84, MSE = 232343126319,5 8 | Gradient Boosting |
| [25], 2023 | Сделки с недвижимостью в центре города Куала-Лумпур, 693 ед. | R 2 = 0.973, Relative Error = 19,4% | Decision Tree | Наиболее успешной моделью является алгоритм Decision Tree с 97% пригодности и 19,4% относительной ошибки. |
| R 2 = 0,900, Relative Error = 41,3% | Случайный лес |
| R 2 = 0,741, Relative Error = 42,1% | Support Vector Machine |
| [26], 2024 | Веб-скраппинг района Кадыкёй в Стамбуле (744 наблюдения) и набор данных домов Kaggle.com (3 000 наблюдений) | R 2 = 0.584745 RMSE = 0.000004 MAPE = 3.851183 | Multiple линейная регрессия | В статье приводятся гибриды каждого алгоритма, но они не являются сферой интересов в рамках исследования. Наибольшую точность показал Support Vector Regression, но и Случайный лес имеет хороший результат |
| R 2 = 0.445784 RMSE = 0.000007 MAPE = 5.561910 | Lasso |
| R 2 = 0.381333 RMSE = 0.000007 MAPE = 5.561910 | Ridge regression |
| R 2 = 0.674176 RMSE = 0.00168 MAPE = 3.421458 | Support Vector Regression (SVR) |
| R 2 = 0.595008 RMSE = 0.000004 MAPE = 4.158058 | AdaBoost |
| R 2 = 0.560765 RMSE = 0.000004 MAPE = 3.939253 | Decision Tree |
| R 2 = 0.624213 RMSE = 0.000003 MAPE = 3.488057 | Случайный лес |
| R 2 = 0.584745 RMSE = 0.000004 MAPE = 3.851183 | XGBoost |

В представленных источниках использовались различные алгоритмы предсказания стоимости, наиболее популярными выступают случайный лес и XGBoost. По результатам рассмотрения и анализа публикаций принято решение в использовании в работе следующих алгоритмов машинного обучения: случайный лес.

* 1. **Алгоритм RandomForestRegressor: принцип работы и особенности**

RandomForestRegressor - это алгоритм машинного обучения, относящийся к классу ансамблевых методов и используемый для задач регрессии. Он является расширением концепции деревьев решений и обладает высокой точностью и устойчивостью к переобучению.

Принцип работы алгоритма включает несколько шагов:

1. Подготовка данных для обучения. Каждая точка данных должна быть представлена набором признаков и соответствующим числовым целевым значением.
2. Генерация деревьев. Алгоритм создаёт деревья по входу, который пользователь указывает параметром n\_estimators. Каждое дерево строится с использованием небольшой случайной группы данных из обучающего набора.
3. Рост деревьев. Каждое дерево строится с использованием меньших групп данных, которые создаются с помощью выбранных признаков. Этот процесс продолжается до тех пор, пока данные нельзя разделить дальше или не будет выполнено определённое условие.
4. Объединение прогнозов. После построения всех деревьев прогнозы объединяются: суммируются прогнозы всех деревьев и вычисляется среднее значение.

Некоторые особенности алгоритма RandomForestRegressor:

1. Работа с большими наборами данных. Алгоритм может эффективно обрабатывать большие наборы данных и многомерные данные без значительного снижения производительности
2. Обработка пропущенных данных. Random Forest может обрабатывать пропущенные значения, сохраняя высокую точность даже с неполными данными
3. Отсутствие необходимости в масштабировании признаков. В отличие от многих других алгоритмов, Random Forest не требует нормализации или масштабирования данных
4. Возможность выбора важных признаков. Это делается путём построения нескольких деревьев решений и измерения важности каждого признака на основе его вклада в точность

Преимущества RandomForestRegressor:

1. Высокая точность: за счет ансамблирования и случайности, RandomForestRegressor обычно дает очень точные прогнозы
2. Устойчивость к переобучению: случайность при построении деревьев помогает предотвратить переобучение на обучающих данных
3. Возможность оценки важности признаков: алгоритм может оценить, какие признаки наиболее важны для предсказания. Это полезно для понимания данных и выбора признаков
4. Простота использования: RandomForestRegressor имеет относительно небольшое количество гиперпараметров, которые нужно настраивать
5. Эффективная работа с большими объемами данных: может обрабатывать большие наборы данных.

Недостатки RandomForestRegressor:

1. Сложность интерпретации: сложно интерпретировать, как именно RandomForestRegressor принимает решения, по сравнению с одним деревом решений
2. Потребление памяти: для хранения всех деревьев в лесу требуется много памяти
3. Время обучения: обучение большого леса может занять много времени
4. Смещение для категориальных признаков с большим количеством значений: если у вас есть категориальные признаки с очень большим количеством уникальных значений, RandomForestRegressor может быть смещен в сторону этих признаков. лучшие результаты.
   1. **Коэффициент детерминации**

Коэффициент детерминации (R2R2 — R-квадрат) — это доля [дисперсии](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D1%81%D0%BF%D0%B5%D1%80%D1%81%D0%B8%D1%8F_%D1%81%D0%BB%D1%83%D1%87%D0%B0%D0%B9%D0%BD%D0%BE%D0%B9_%D0%B2%D0%B5%D0%BB%D0%B8%D1%87%D0%B8%D0%BD%D1%8B) зависимой переменной, объясняемая рассматриваемой [моделью](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C) зависимости, то есть объясняющими переменными. Более точно — это единица минус доля необъяснённой дисперсии (дисперсии случайной ошибки модели, или условной по факторам дисперсии зависимой переменной) в дисперсии зависимой переменной. Его рассматривают как универсальную меру зависимости одной случайной величины от множества других. В частном случае линейной зависимости R2является квадратом так называемого [множест-венного коэффициента корреляции](https://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D1%8D%D1%84%D1%84%D0%B8%D1%86%D0%B8%D0%B5%D0%BD%D1%82_%D0%BA%D0%BE%D1%80%D1%80%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%B8) между зависимой переменной и объясняю-щими переменными. В частности, для модели парной линейной регрессии коэффициент детерминации равен квадрату обычного коэффициента корреляции между *y* и *x*.

Основная проблема применения (выборочного) R2R2 заключается в том, что его значение увеличивается (не уменьшается) от добавления в модель новых переменных, даже если эти переменные никакого отношения к объясняемой переменной не имеют. Поэтому сравнение моделей с разным количеством факторов с помощью коэффициента детерминации, вообще говоря, некорректно. Для этих целей можно использовать альтернативные показатели.

Истинный коэффициент детерминации модели зависимости случайной величины *y* от факторов *x* определяется следующим образом:

Где D[y]=– дисперсия случайной величины y, а D[y|x] = – условная (по факторам x) дисперсия зависимой переменной (дисперсия ошибки модели). В данном случае используются истинные параметры, характеризующие распределение случайных величин.

Коэффициент детерминации для модели с константой принимает значения от 0 до 1. Чем ближе значение коэффициента к 1, тем сильнее зависимость. При оценке регрессионных моделей это интерпретируется как соответствие модели данным. Для приемлемых моделей предполагается, что коэффициент детерминации должен быть хотя бы не меньше 50 % (в этом случае коэффициент множественной корреляции превышает по модулю 70 %). Модели с коэффициентом детерминации выше 80 % можно признать достаточно хорошими (коэффициент корреляции превышает 90 %). Значение коэффициента детерминации 1 означает функциональную зависимость между переменными.

1. **Архитектура и реализация веб-приложения**
   1. **Обзор архитектуры Model-View-Template (MVT)**

Фреймворк Django реализует архитектурный паттерн Model-View-Template или сокращенно MVT, который по факту является модификацией распространённого в веб-программировании паттерна MVC (Model-View-Controller).

Она разделяет приложение на три основных компонента:

1. **Model (Модель):** отвечает за управление данными, хранение и обработку информации. Модель определяет структуру данных, методы доступа к ним, логику валидации и взаимодействия с базой данных. В MVT модели обычно представляют собой классы Python, наследуемые от базовых классов Django, и описывают поля данных, типы данных, отношения между данными и бизнес-логику.
2. **View (Представление):** отвечает за обработку пользовательских запросов, взаимодействие с моделями и передачу данных в шаблоны. Представления принимают запросы от пользователей, обрабатывают логику приложения, взаимодействуют с моделями для получения или изменения данных, а затем выбирают соответствующий шаблон и передают ему данные для отображения пользователю. В Django представления обычно являются функциями или классами, принимающими запрос HTTP и возвращающими ответ HTTP.
3. **Template (Шаблон):** отвечает за отображение данных пользователю. Шаблон содержит статический HTML-код и элементы, позволяющие динамически отображать данные, полученные от представления. Он включает в себя переменные, логику (например, циклы и условные операторы) и фильтры для обработки данных. Шаблоны отделены от логики приложения, что обеспечивает разделение представления и логики, делая код более читаемым и удобным для поддержки. В Django шаблоны написаны на собственном шаблонизаторе Django Template Language (DTL).

Архитектуру MVT изображена на рисунке 1.

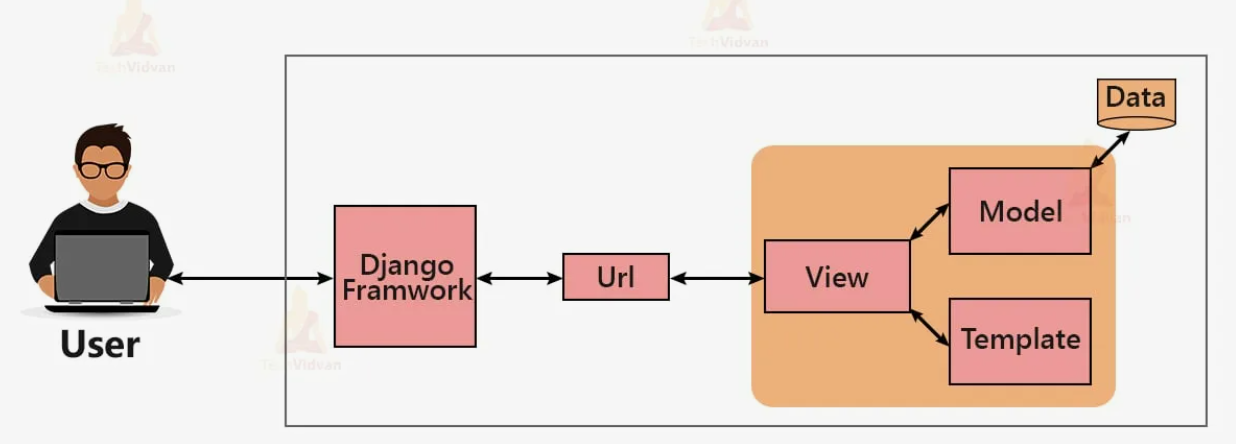


Рис.1 - Архитектура Django (MVT)

**3. Программная реализация**

* 1. **Описание функциональности приложения**

На рисунке 2 изображена часть датасета, на котором проводилось обучение:

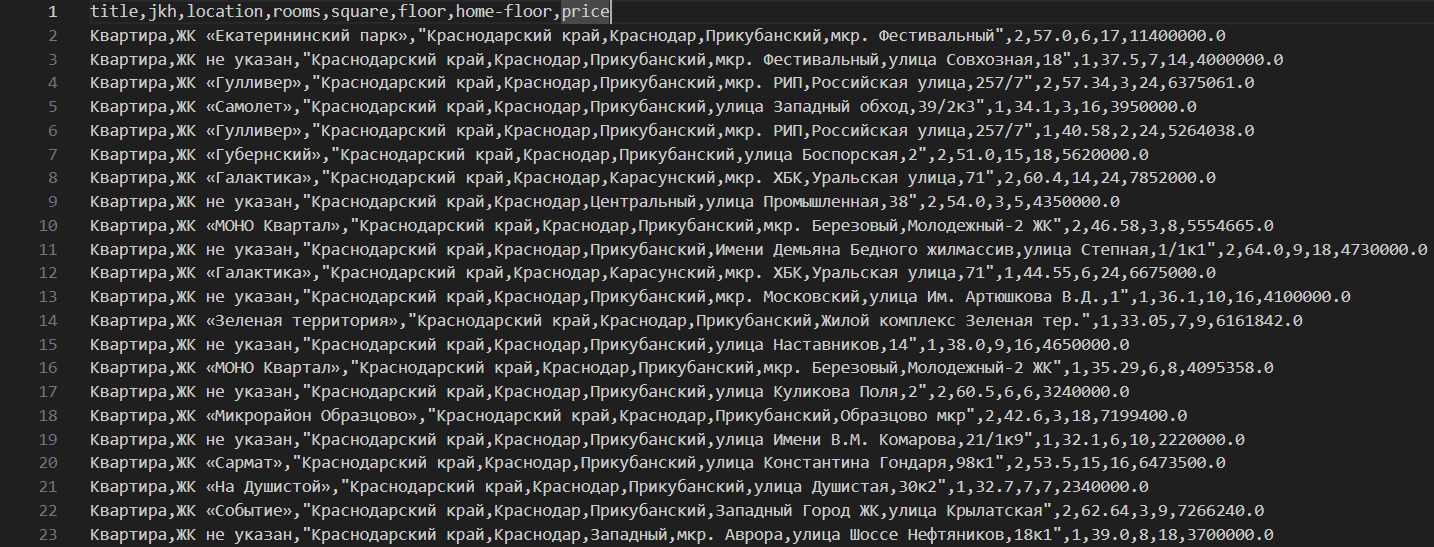


Рис.2 – датасет

Предикторы: title (название), jkh (ЖК), location (адрес), rooms (количество комнат), square (площадь квартиры), floor (этаж), home-floor (этажность дома), price (цена в рублях).

На рисунке 3 изображен интерфейс приложения. Приложение для предсказания цены квартиры предоставляет пользователю следующий функционал:

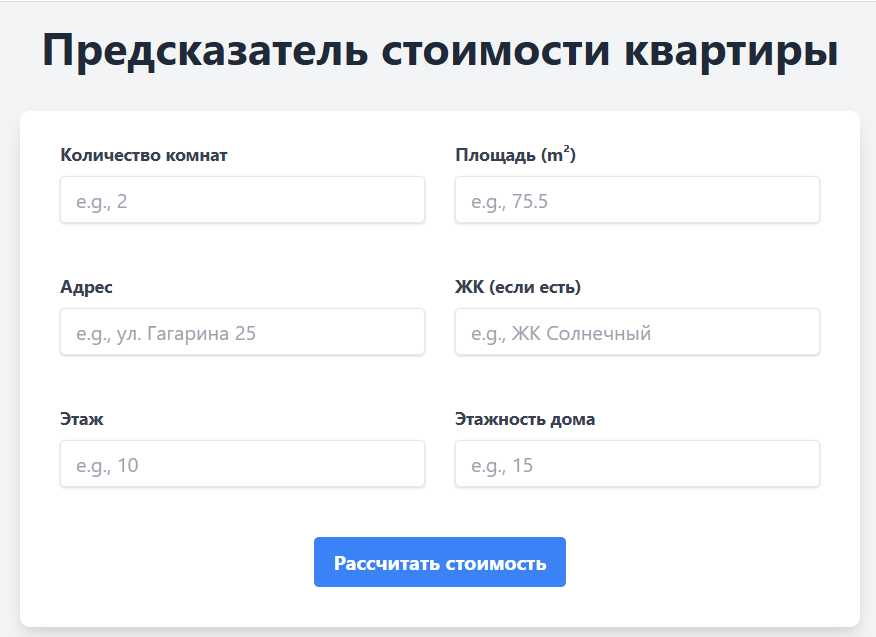


Рис.3 – интерфейс приложения

**4 Исследовательская часть**

Рассмотрим работу приложения при разных входных данных и сравним результаты.

**4.1 Результаты работы приложения**

На рисунках 4 и 6 представлены исходные данные, а на рисунках 5 и 7 – результат работы программы (предсказанная цена)

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

Рис.4 – исходные данные

Изображение выглядит как текст, снимок экрана, Шрифт, Бренд

Контент, сгенерированный ИИ, может содержать ошибки.

Рис.5 – результат

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

Рис.6 – исходные данные

Изображение выглядит как текст, снимок экрана, Шрифт, Бренд

Контент, сгенерированный ИИ, может содержать ошибки.

Рис.7 – результат

Так как используется алгоритм RandomForestRegressor, то выводим первое дерево леса. Дерево изображено на рисунке 8.

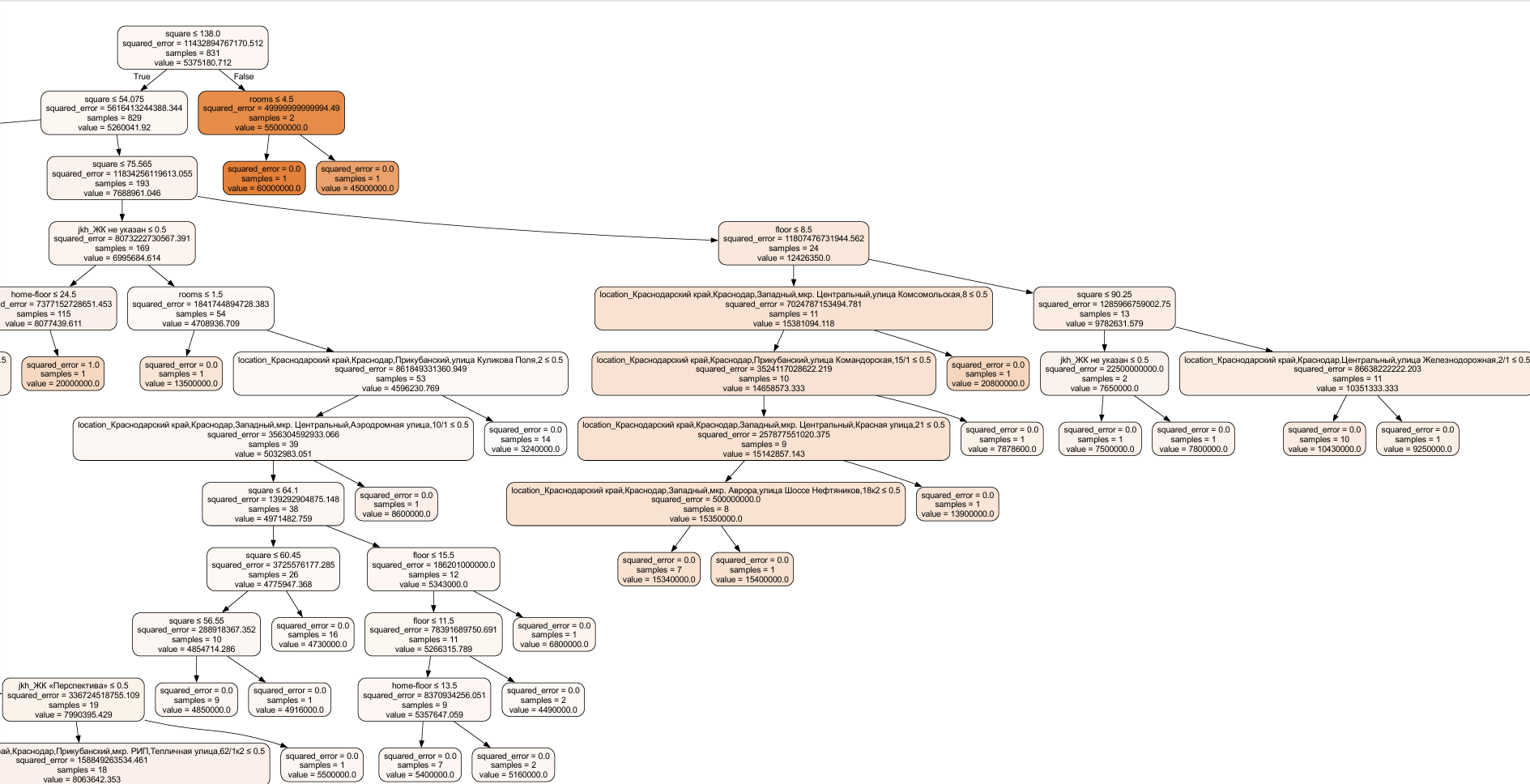


Рис.8 – фрагмент первого дерева леса

График на рисунке 9 показывает, насколько хорошо модель RandomForestRegressor предсказывает значения. Ось абсцисс представляет собой предсказанные значения моделью. То есть, какие значения модель считает правильными для каждого объекта. Ось ординат представляет собой фактические значения. Метрика R2 имеет значение 82%, это означает, что наша модель RandomForestRegressor объясняет примерно 82% изменчивости в целевой переменной (то есть, в ценах на квартиры), что достаточно хороший показатель.

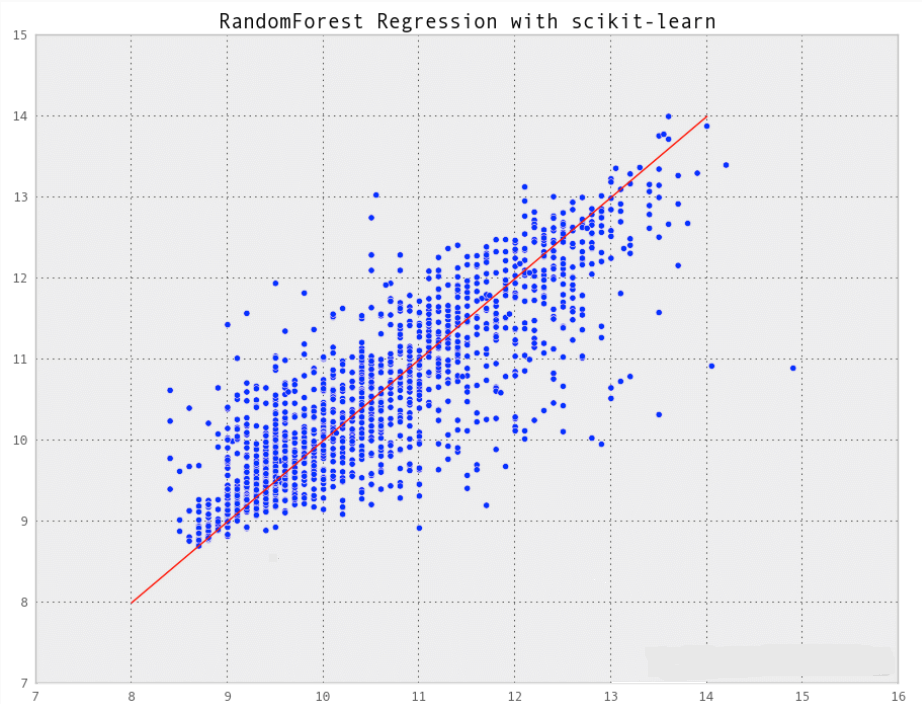


Рис.9 – график рассеивания

**ЗАКЛЮЧЕНИЕ**

В заключение, данная курсовая работа была посвящена разработке приложения для предсказания цены квартиры, основанного на алгоритмах машинного обучения. Были рассмотрены ключевые аспекты создания подобного решения, начиная от сбора и предварительной обработки данных, заканчивая выбором и обучением модели машинного обучения, а также проектированием пользовательского интерфейса.

В ходе работы:

* Проведен анализ предметной области и определены факторы, влияющие на цену квартиры.
* Разработан алгоритм для вычисления предсказанной стоимости квартиры на основе выбранных параметров.
* Использован метод машинного обучения RandomForestRegressor для построения модели предсказания цены.
* Реализован пользовательский интерфейс, обеспечивающий удобный ввод данных и отображение результатов.
* Введена оценка адекватности, позволяющая оценить надежность предсказания.

Результаты работы показывают, что машинное обучение может быть о ценах, рекомендации и интеграцию с другими сервисами.

В целом, данная курсовая работа представляет собой ценный вклад в разработку интеллектуальных систем оценки недвижимости и может служить основой для дальнейших исследований в этой области. Результаты, полученные в ходе работы, могут быть использованы на практике как частными лицами, так и профессиональными участниками рынка недвижимости.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

* 1. Matplotlib — Visualization with Python. – URL: https://matplotlib.org/ – Текст: электронный.
  2. Методы машинного обучения в исследовании рынка жилой недвижимости. – URL: https://cyberleninka.ru/article/n/metody-mashinnogoobucheniya-v-issledovanii-rynka-zhiloy-nedvizhimosti/viewer – Текст: электронный.
  3. NumPy Documentation. – URL: https://numpy.org/doc/ – Текст: электронный.
  4. Pandas Tutorial – W3Schools offers free online tutorials, references and exercises in all the major languages of the web. Covering popular subjects like HTML, CSS, JavaScript, Python, SQL, Java, and many, many more. – URL: https://www.w3schools.com/python/pandas/default.asp – Текст: электронный.
  5. Анализ трендов рынка недвижимости: методы и подходы. – URL: https://cyberleninka.ru/article/n/analiz-trendov-rynka-nedvizhimosti-metody-ipodhody/viewer – Текст: электронный.
  6. Массовая оценка объектов недвижимости на основе технологий машинного обучения. Анализ точности различных методов на примере определения рыночной стоимости квартир. – URL: https://cyberleninka.ru/article/n/massovaya-otsenka-obektov-nedvizhimosti-naosnove-tehnologiy-mashinnogo-obucheniya-analiz-tochnosti-razlichnyh-metodovna-primere/viewer – Текст: электронный.
  7. Neural network models (supervised) — scikit-learn 1.5.0 documentation. – URL: https://scikit-learn.org/stable/modules/neural\_networks\_supervised.html – Текст: электронный.
  8. **Malpezzi, S.** Hedonic Pricing Models: A Selective and Applied Review / S. Malpezzi // Housing Economics and Public Policy / eds. T. O’Sullivan, K. Gibb. Wiley, 2002. Hedonic Pricing Models. – 67-89 p.
  9. Fuzzy Logic Toolbox Documentation. – URL: https://www.mathworks.com/help/fuzzy/ – Текст: электронный.
  10. Метод опорных векторов SVM - scikit-learn. – URL: https://scikitlearn.ru/1-4-support-vector-machines/ – Текст: электронный.
  11. Линейная регрессия — Документация ML Cheatsheet. – URL: https://ml-cheatsheet-russian.readthedocs.io/ru/latest/linear\_regression.html – Текст: электронный.
  12. **Su, J.** A Fast Decision Tree Learning Algorithm – URL: https://fileadmin.cs.lth.se/ai/Proceedings/aaai06/05/AAAI06-080.pdf – Текст: электронный.
  13. RandomForestRegressor — scikit-learn 1.5.0 documentation. – URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForest Regressor.html – Текст: электронный.
  14. **Jivani, A.** The Adept K-Nearest Neighbour Algorithm - An optimization to the Conventional K-Nearest Neighbour Algorithm // Transactions on Machine Learning and Artificial Intelligence. 2016. Т. 4.
  15. PLSRegression – Gallery examples: Compare cross decomposition methods Principal Component Regression vs Partial Least Squares Regression. – URL: https://scikit-learn/stable/modules/generated/sklearn.cross\_decomposition. PLSRegression.html – Текст: электронный.
  16. **Zhang, H.** The Optimality of Naive Bayes – URL: https://www.cs.unb.ca/~hzhang/publications/FLAIRS04ZhangH.pdf – Текст: электронный.
  17. Getting started with Multivariate Multiple Regression | UVA Library. – URL: https://library.virginia.edu/data/articles/getting-started-with-multivariatemultiple-regression – Текст: электронный.
  18. Introduction to spatial analysis | Documentation | ArcGIS Developers. – URL: https://developers.arcgis.com/documentation/mapping-apis-andservices/spatial-analysis/ – Текст: электронный.
  19. XGBoost Documentation — xgboost 2.0.3 documentation. – URL: https://xgboost.readthedocs.io/en/stable/ – Текст: электронный.
  20. Lasso – Gallery examples: Release Highlights for scikit-learn 1.4 Release Highlights for scikit-learn 0.23 Compressive sensing: tomography reconstruction with L1 prior (Lasso) Joint feature selection with ... – URL: https://scikitlearn/stable/modules/generated/sklearn.linear\_model.Lasso.html – Текст: электронный.
  21. Ensemble methods — scikit-learn 0.16.1 documentation. – URL: https://scikit-learn.org/0.16/modules/ensemble.html – Текст: электронный.
  22. Статистический анализ динамики стоимости квартир на вторичном рынке жилой недвижимости города Москвы. – URL: https://cyberleninka.ru/article/n/statisticheskiy-analiz-dinamiki-stoimosti-kvartirna-vtorichnom-rynke-zhiloy-nedvizhimosti-goroda-moskvy/viewer – Текст: электронный.
  23. **Ясницкий, Л. Н.** Разработка и применение комплексных нейросетевых моделей массовой оценки и прогнозирования стоимости жилых объектов на примере рынков недвижимости Екатеринбурга и Перми / Л.Н. Ясницкий, Я. В.Л. – Текст: непосредственный. // Имущественные отношения в Российской Федерации. 2017. № 3 (186). – С. 68-84.
  24. **Агафонова, Е. О.** Сравнение эффективности методов машинного обучения в задаче оценки стоимости недвижимости / Е.О. Агафонова, А.А. Белоусов // Информационные технологии и нанотехнологии (ИТНТ-2023) : сб. тр. по материалам IX Междунар. конф. и молодеж. шк. – г. Самара: Изд-во Самар. ун-та, 2023Т. 5: Науки о данных, 2023.
  25. **Mohd, T.** Office Rent Prediction based on the Influenced Features // Environment-Behaviour Proceedings Journal. 2022. Vol. 7. № 19. – 61-68 p.
  26. **Özöğür Akyüz, S. A** Novel Hybrid House Price Prediction Model // Computational Economics. 2023. Vol. 62. № 3. – 1215-1232 p.

**ПРИЛОЖЕНИЕ А**

Файл views.py:

from django.shortcuts import render

import pandas as pd

import numpy as np

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

from sklearn.tree import export\_graphviz

import graphviz

from sklearn.tree import DecisionTreeRegressor

import random

def my\_r2\_score(y\_true, y\_pred):

    y\_true = np.array(y\_true)

    y\_pred = np.array(y\_pred)

    y\_mean = np.mean(y\_true)

    sst = np.sum((y\_true - y\_mean) \*\* 2)

    sse = np.sum((y\_true - y\_pred) \*\* 2)

    if sst == 0:  # Handle the case where all y\_true values are the same

        return 1.0 if np.all(y\_true == y\_pred) else 0.0  # If y\_true and y\_pred are also the same, R2 = 1, otherwise 0

    r2 = 1 - (sse / sst)

    return r2

class MyRandomForestRegressor:

    def \_\_init\_\_(self, n\_estimators=100, max\_depth=None, random\_state=None):

        self.n\_estimators = n\_estimators

        self.max\_depth = max\_depth

        self.random\_state = random\_state

        self.trees = []

        if self.random\_state is not None:

            random.seed(self.random\_state)

    def \_bootstrap\_sample(self, X, y):

        n\_samples = X.shape[0]

        indices = np.random.choice(n\_samples, n\_samples, replace=True)

        return X.iloc[indices], y.iloc[indices]

    def fit(self, X, y):

        self.feature\_names = list(X.columns)

        self.trees = []

        for \_ in range(self.n\_estimators):

            X\_sample, y\_sample = self.\_bootstrap\_sample(X, y)

            tree = DecisionTreeRegressor(max\_depth=self.max\_depth, random\_state=random.randint(0, 1000))  # Each tree gets a unique random\_state

            tree.fit(X\_sample, y\_sample)

            self.trees.append(tree)

    def predict(self, X):

        predictions = np.array([tree.predict(X) for tree in self.trees])

        return np.mean(predictions, axis=0)  # Average the predictions of all trees

try:

    data = pd.read\_csv('krasnodar\_apartments\_cleaned.csv')

    print("Data loaded successfully. Shape:", data.shape)

except FileNotFoundError:

    print("Error: krasnodar\_apartments\_cleaned.csv not found.")

    data = pd.DataFrame()

except Exception as e:

    print(f"Error loading CSV: {e}")

    data = pd.DataFrame()

if not data.empty:

    for col in ['rooms', 'square', 'floor', 'home\_floor']:

        if col in data.columns:

            data[col] = pd.to\_numeric(data[col], errors='coerce')

            data[col] = data[col].fillna(data[col].median())

    data = pd.get\_dummies(data, columns=['location', 'jkh'], drop\_first=True)

    X = data.drop(['price', 'id', 'title'], axis=1, errors='ignore')

    y = data['price']

    missing\_cols = set(X.columns) - set(['rooms', 'square', 'floor', 'home\_floor'])

    for c in missing\_cols:

        X[c] = 0

    try:

        X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

        model = MyRandomForestRegressor(n\_estimators=100, random\_state=42, max\_depth=10)  # Use our custom implementation and set max\_depth

        model.fit(X\_train, y\_train)

        y\_pred = model.predict(X\_test)

        r2 = my\_r2\_score(y\_test, y\_pred) \* 100

        feature\_names = list(X.columns) # Store feature names for later use

    except ValueError as e:

        print(f"Error during model training: {e}")

        X\_train, X\_test, y\_train, y\_test, model, r2, feature\_names = None, None, None, None, None, 0, []

    except Exception as e:  # Catch other potential errors

        print(f"An unexpected error occurred during model training: {e}")

        X\_train, X\_test, y\_train, y\_test, model, r2, feature\_names = None, None, None, None, None, 0, []

else:

    print("No data loaded, skipping model training.")

    X\_train, X\_test, y\_train, y\_test, model, r2, feature\_names = None, None, None, None, None, 0, []

def predict\_price(input\_data):

    global model, feature\_names

    if model is None:  # Check that the model is trained

        print("Error: Model not trained.")

        return None

    input\_df = pd.DataFrame([input\_data])

    input\_df = pd.get\_dummies(input\_df, columns=['location', 'jkh'], drop\_first=True)

    input\_df = input\_df.reindex(columns=X.columns, fill\_value=0)

    if len(input\_df.columns) != len(X.columns):

        print("Error: Input data columns do not match training data columns.")

        return None

    predicted\_price\_rub = model.predict(input\_df)[0]

    return predicted\_price\_rub

def home(request):

    global model, r2, feature\_names

    if model is not None:

        try:

            tree = model.trees[0]  # Access the first tree in the forest

            dot\_data = export\_graphviz(tree,

                                        feature\_names=feature\_names,  # Pass feature names

                                        filled=True,

                                        rounded=True,

                                        special\_characters=True)

            graph = graphviz.Source(dot\_data)

            graph.render("flat\_price\_tree", view=False)

        except Exception as e:

            print(f"Error generating tree visualization: {e}")

    else:

        print("Model not trained, skipping tree visualization.")

    if request.method == 'POST':

        try:

            input\_data = {

                'rooms': int(request.POST.get('rooms')),

                'square': float(request.POST.get('square')),

                'location': request.POST.get('location'),

                'jkh': request.POST.get('jkh', ''),

                'floor': int(request.POST.get('floor')),

                'home\_floor': int(request.POST.get('home\_floor')),

            }

        except ValueError:  # Handle potential errors from user input

            return render(request, 'prediction/result.html', {

                'predicted\_price': "Ошибка: Неверный ввод данных",

                'r2': round(r2, 2) if model is not None else 0,

            })

        if model is not None:  # Check that the model is trained

            predicted\_price = predict\_price(input\_data)

        else:

            predicted\_price = None

        return render(request, 'prediction/result.html', {

            'predicted\_price': round(predicted\_price, 2) if predicted\_price is not None else "Ошибка: Модель не обучена",

            'r2': round(r2, 2) if model is not None else 0,

        })

    return render(request, 'prediction/home.html', {'r2': round(r2, 2) if model is not None else 0})