**МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА РУТ (МИИТ)**

Высшая инженерная школа

*КУРСОВАЯ РАБОТА*

По дисциплине: «Машинное обучение и анализ данных»

Тема: «Построение модели прогнозирования времени доставки грузов для железнодорожных перевозок (местный тип парка)»

|  |  |
| --- | --- |
|  | **Выполнил:**  Студент (-ка) ШЦТ-211  Артемова А.Д. |
| Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  «\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_202\_\_г.  \_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) (расшифровка) | **Руководитель**:  Ахметова Р.А. |
|  |  |

Москва 2025 г.

Содержание

[Описание задачи 3](#_Toc199330361)

[Основная часть 4](#_Toc199330362)

[1. Разведочный анализ данных (EDA) 4](#_Toc199330363)

[2. Подготовка данных 6](#_Toc199330364)

[3. Анализ отобранных фич (EDA) 9](#_Toc199330365)

[4. Генерация признаков 12](#_Toc199330366)

[4.1 Создание новых признаков 12](#_Toc199330367)

[4.2 Кодирование категориальных переменных 14](#_Toc199330368)

[4.3 Устранение выбросов 14](#_Toc199330369)

[5. Разделение на выборки 15](#_Toc199330370)

[6. Построение модели 15](#_Toc199330371)

[6.1 Baseline-модель с использованием AutoML 15](#_Toc199330372)

[6.2 Классическое моделирование 16](#_Toc199330373)

[7. Лучшая модель 19](#_Toc199330374)

[8. Интерпретация лучшей модели 20](#_Toc199330375)

[8.1 Глобальная интерпретация 20](#_Toc199330376)

[8.2 Локальная интерпретация 23](#_Toc199330377)

[9. Дополнительные решения по работе 24](#_Toc199330378)

[9.1 Permutation Feature Importance 24](#_Toc199330379)

[9.2 Outlier Detection — Isolation Forest 24](#_Toc199330380)

[9.3 Dummy Regressor (базовая модель для сравнения) 25](#_Toc199330381)

[9.4 Собственная метрика качества: MAE по праздничным дням 26](#_Toc199330382)

[Выводы 27](#_Toc199330383)

# **Описание задачи**

В данной курсовой работе рассматривается задача прогнозирования одной из ключевых характеристик процесса перевозки грузов — фактического срока доставки. Данные для анализа представляют собой набор реальных наблюдений, содержащих информацию об отправке грузов железнодорожным транспортом. В качестве целевой переменной используется показатель «Срок доставки факт», то есть фактическое время, затраченное на перевозку конкретной отправки от пункта отправления до пункта назначения.

С точки зрения машинного обучения, поставленная задача относится к классу задач регрессии, поскольку требуется предсказать количественное значение целевой переменной на основании множества признаков, описывающих параметры отправления. Среди этих признаков представлены как категориальные, так и числовые характеристики, включая сведения о типе груза, операторе перевозки, расстоянии, различных временных метках, типе парка и другие факторы, влияющие на логистику процесса.

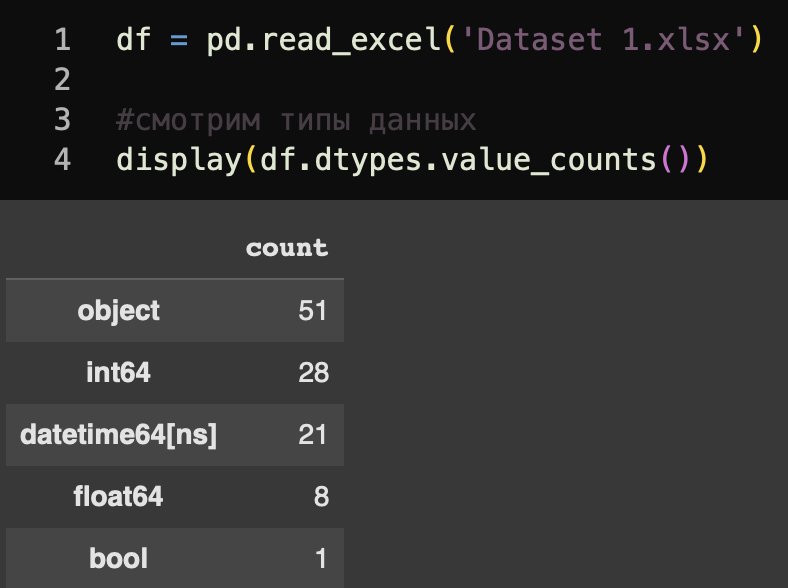
В рамках работы основное внимание уделяется построению и сравнению различных моделей машинного обучения для достижения наилучших прогнозных характеристик. В качестве ключевых метрик качества моделей планируется использовать среднюю абсолютную ошибку (MAE), среднеквадратичную ошибку (MSE) и коэффициент детерминации (R²), что позволит всесторонне оценить точность и пригодность разработанных решений для практического применения в логистической сфере. Анализ будет включать в себя исследование структуры и распределения данных, оценку влияния различных признаков на целевую переменную, а также интерпретацию полученных моделей с целью выявления факторов, наиболее существенно влияющих на сроки доставки.

Работа реализуется на основе данных с ограничением по типу парка, как это предусмотрено вариантом задания. Все этапы — от предобработки и анализа данных до построения и оценки моделей — будут подробно рассмотрены с опорой на реальные вычисления и визуализации, выполненные в Google Colab.

# **Основная часть**

## **1. Разведочный анализ данных (EDA)**

На первом этапе работы был проведён разведочный анализ исходного набора данных, что позволило сформировать представление о структуре и качестве имеющейся информации. Исходные данные были загружены из файла «Dataset 1.xlsx», после чего выполнен первичный осмотр типов данных (рис. 1).



*Рисунок 1. Загрузка данных*

Анализ показал, что в датасете присутствуют 51 категориальный признак (тип object), 28 целочисленных признаков (int64), 21 признак с типом данных datetime, 8 признаков с плавающей точкой (float64), а также один булевый признак. Такое распределение типов данных свидетельствует о том, что датасет содержит как качественные, так и количественные переменные, а значительная часть информации представлена в виде категориальных признаков, что определяет необходимость дальнейшей их обработки и кодирования на этапах подготовки данных.

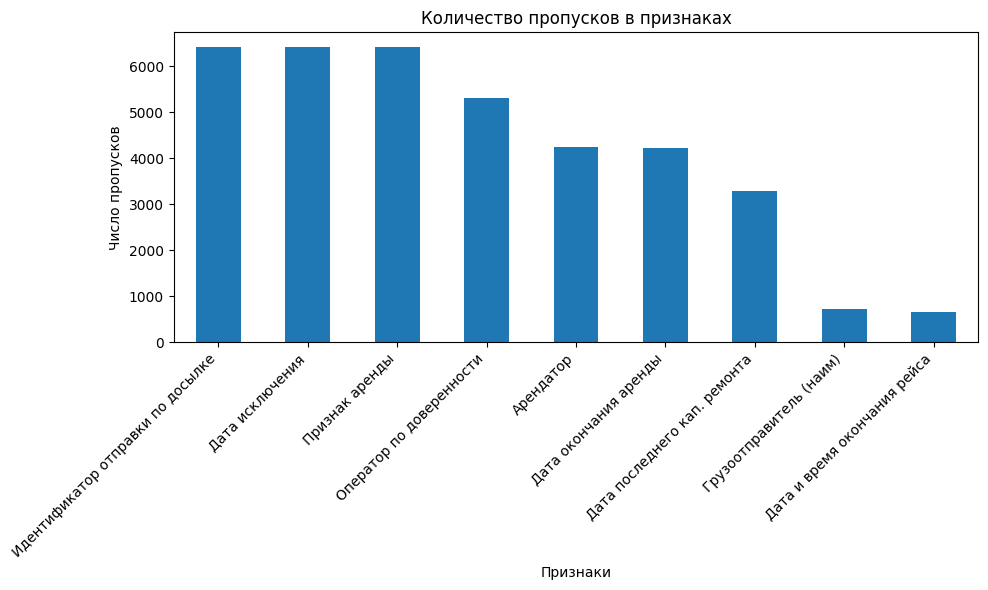
Далее, в соответствии с условиями задания, была осуществлена фильтрация по признаку «Тип парка», для отбора только тех записей, где содержится слово «Местный» (листинг 1).

*Листинг 1. Фильтрация по признаку*

dff = df[df['Тип парка'].astype(*str*).str.contains("Местный", case=*False*, na=*False*)].copy()  
  
*print*(f"до фильтра:{df.shape}")  
*print*(f"после фильтра:{dff.shape}")

После применения фильтра объём данных существенно сократился: если до фильтрации в таблице содержалось 12 093 строки и 109 столбцов, то после фильтрации осталось 6 413 строк при неизменном количестве признаков. Это говорит о том, что доля записей, относящихся к интересующему типу парка, составляет чуть более половины исходного объёма.

На следующем этапе для каждого столбца было подсчитано количество отсутствующих значений, что позволило выявить наиболее проблемные переменные. Результаты анализа визуализированы на столбчатой диаграмме, отражающей распределение числа пропусков по признакам (рис. 2).



*Рисунок 2. Количество пропусков в признаках*

Особенно большое количество пропусков наблюдается в признаках «Идентификатор отправки по досылке», «Дата исключения», «Признак аренды», «Оператор по доверенности» и некоторых других. Причём для первых трёх признаков количество пропусков в принципе равно количеству строк, следовательно, эти столбцы удаляются.

Важной задачей на этапе разведочного анализа стало исследование распределения некоторых количественных признаков, непосредственно связанных с особенностями процесса перевозки. Например, анализ распределения времени простоя («Простой») был выполнен с использованием boxplot-графика (рис. 3).



*Рисунок 3. Распределение простоев*

В результате стало очевидно, что распределение данного признака характеризуется наличием большого количества выбросов, то есть наблюдаются отдельные случаи аномально долгого простоя.

Таким образом, проведённый первичный анализ данных позволил выявить основные проблемы датасета — наличие большого числа пропусков и выбросов, а также слабую взаимосвязь между большинством числовых признаков и целевой переменной. При этом среди всех переменных наибольший интерес представляет «Расстояние общее» как один из наиболее информативных факторов.

## **2. Подготовка данных**

На этапе подготовки данных была проведена их очистка и первичная трансформация с целью улучшения качества будущего моделирования. В первую очередь были удалены дубликаты и строки с пустыми значениями в целевой переменной (листинг 2).

*Листинг 2. Очистка целевой переменной*

dff = dff.drop\_duplicates().reset\_index(drop=*True*)  
  
dff = dff[~dff['Срок доставки факт'].isna()].copy()

На следующем этапе был проведён осознанный выбор признаков, которые будут использоваться для построения моделей. Признаки были разделены на три основных группы: временные, числовые и категориальные, отдельно была выделена целевая переменная (листинг 3).

*Листинг 3. Выбор признаков*

date = ['Нормативный срок доставки', 'Дата начала', 'Дата завершения']  
numerics = ['Вес груза (кг)', 'Расстояние общее (км)', 'Пробег общий (км)']  
categorical = ['Дорога отправления', 'Дорога назначения', 'Государство отправления']  
target = 'Срок доставки факт'

Отдельно стоит отметить признак «Вес груза (кг)». Несмотря на то, что он содержит значительное количество нулевых значений, данный признак был включён в анализ, чтобы впоследствии с помощью модели определить, влияет ли данная информация на срок доставки.

Далее, все временные признаки были приведены к единому формату datetime для последующего корректного использования при генерации новых признаков. На основе выбранных признаков была создана рабочая витрина данных для моделирования, в которую вошли временные, числовые и категориальные переменные, а также целевая переменная (листинг 4).

*Листинг 4. Рабочая витрина данных для моделирования*

dff['Нормативный срок доставки'] = pd.to\_datetime(dff['Нормативный срок доставки'])  
dff['Дата начала'] = pd.to\_datetime(dff['Дата начала'])  
dff['Дата завершения'] = pd.to\_datetime(dff['Дата завершения'])  
  
dfv = dff[date + numerics + categorical + [target]].copy()

На этапе генерации новых признаков были рассчитаны дополнительные переменные, отражающие особенности перевозок (листинг 5). В частности, был рассчитан признак «Нормативное количество дней доставки» как разница между нормативным сроком доставки и датой начала, признак «Месяц отправки» (номер месяца), «День недели отправки», бинарный признак «Выходной день» (отдельно отмечены выходные дни).

*Листинг 5. Расчет новых переменных (1)*

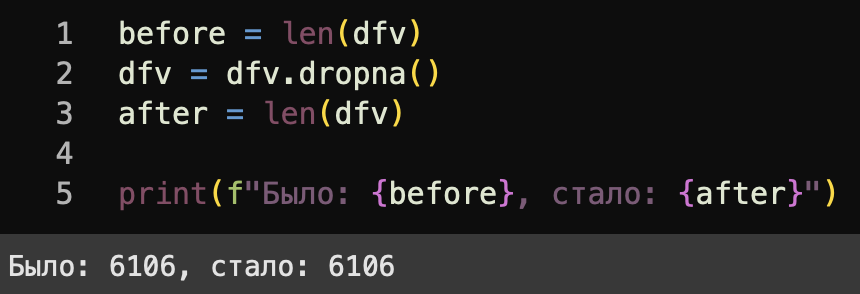
*#расчет новых фич*dfv['Нормативное количество дней доставки'] = (dfv['Нормативный срок доставки'] - dfv['Дата начала']).dt.days  
dfv['Месяц отправки'] = dfv['Дата начала'].dt.month  
dfv['День недели отправки'] = dfv['Дата начала'].dt.weekday + 1  
dfv['Выходной день'] = dfv['День недели отправки'].isin([6, 7]).astype(*int*) *#бинарный*

А также был рассчитан признак «Время года», определяемый по месяцу отправки. Для расчёта времени года использовалась собственная функция, где зима соответствует 1, весна — 2, лето — 3, осень — 4 (листинг 6).

*Листинг 6. Расчет новых переменных (2)*

*def* assign\_season(month):  
 *if* month *in* [12, 1, 2]:  
 *return* 1  
 *elif* month *in* [3, 4, 5]:  
 *return* 2  
 *elif* month *in* [6, 7, 8]:  
 *return* 3  
 *elif* month *in* [9, 10, 11]:  
 *return* 4  
  
dfv['Время года'] = dfv['Месяц отправки'].apply(assign\_season) *#1-зима...4-осень*numeric = numerics + [  
 'Нормативное количество дней доставки',  
 'Месяц отправки',  
 'День недели отправки',  
 'Выходной день',  
 'Время года'  
]

После формирования новых признаков из рабочей витрины были удалены строки с пропущенными значениями, чтобы гарантировать полноту данных для моделирования (рис. 4).



*Рисунок 4. Удаление пропущенных значений*

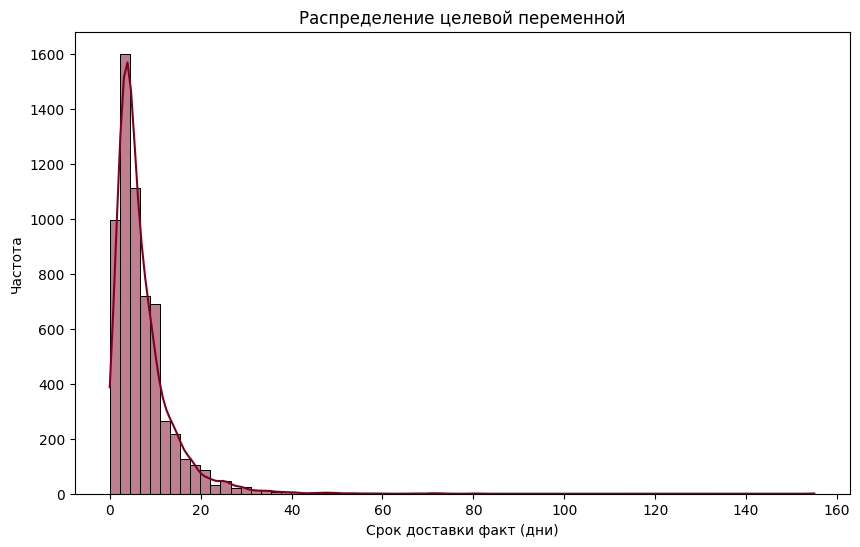
На заключительном этапе подготовки данных была создана итоговая витрина данных, структура которой представлена в виде таблицы 1.

*Таблица 1*

|  |  |
| --- | --- |
| **Тип признаков** | **Значения** |
| Числовые признаки | Вес груза (кг), Расстояние общее (км), Пробег общий (км), Нормативное количество дней доставки, Месяц отправки, День недели отправки, Выходной день, Время года |
| Категориальные признаки | Дорога отправления, Дорога назначения, Государство отправления |
| Целевая переменная | Срок доставки факт |
| Размер витрины | 6106, 15 |

## **3. Анализ отобранных фич (EDA)**

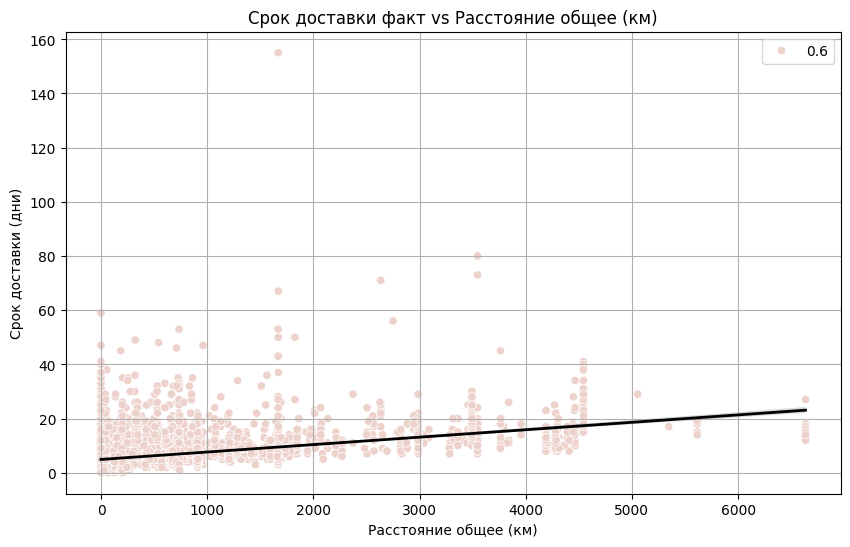
Первым шагом была построена гистограмма распределения целевой переменной — «Срок доставки факт» (рис. 5).



*Рисунок 5. Распределение целевой переменной*

Построенный график позволяет увидеть, что распределение срока доставки имеет выраженную асимметрию и смещено влево: основная масса перевозок завершается в интервале примерно от 3 до 10 дней. При этом на графике фиксируются отдельные случаи экстремально долгой доставки (более 40 дней), что указывает на присутствие выбросов и аномальных ситуаций в данных.

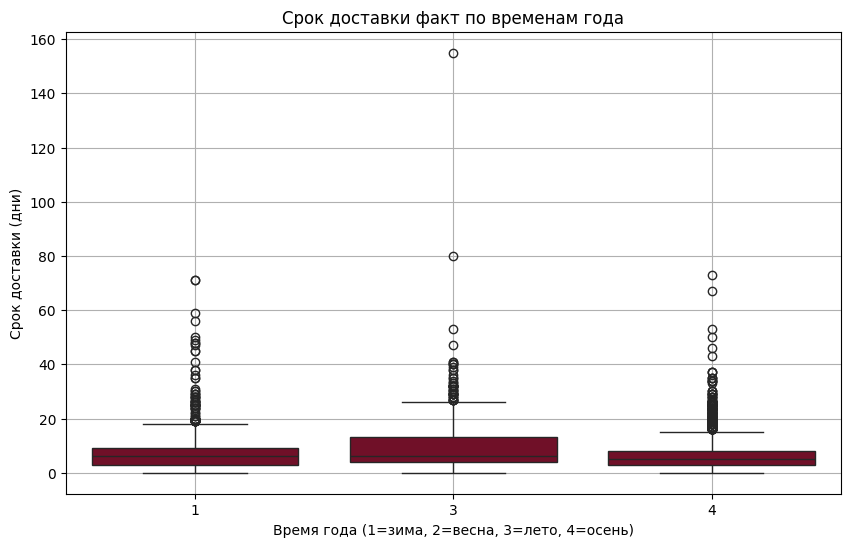
Далее, для изучения взаимосвязи между сроком доставки и расстоянием перевозки был построен график зависимости срока доставки от общего расстояния, что позволяет проанализировать линейные и нелинейные закономерности (рис. 6).



*Рисунок 6. Зависимость срока доставки факт от общего расстояния*

На данном графике визуально определяется слабая положительная корреляция между расстоянием перевозки и сроком доставки: с увеличением расстояния в среднем возрастает и срок. Однако разброс точек остаётся существенным, что говорит о наличии дополнительных факторов, влияющих на сроки доставки, помимо одного лишь расстояния.

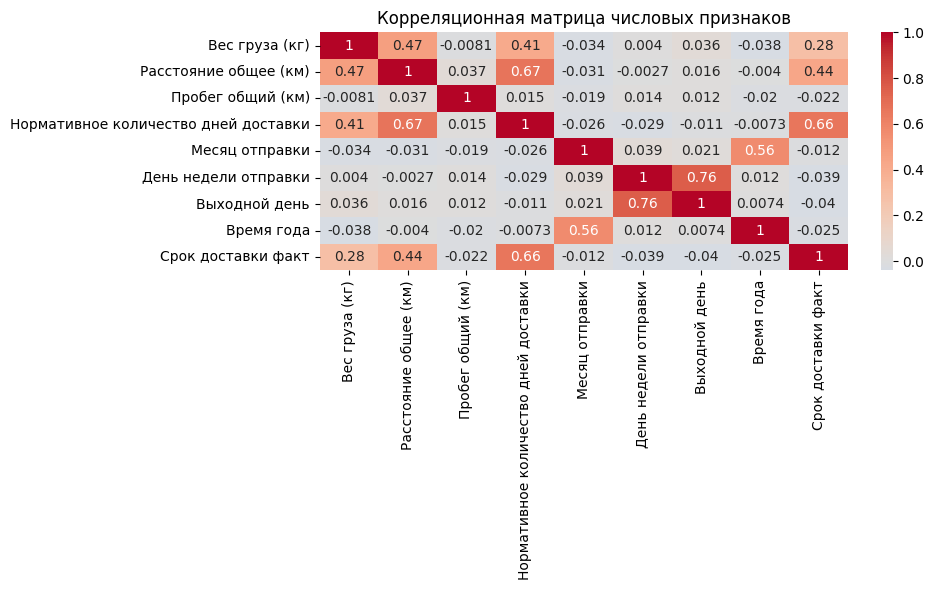
Для выявления сезонных эффектов в данных был построен boxplot, отражающий распределение сроков доставки по временам года (рис. 7).



*Рисунок 7. Срок доставки факт по временам года*

По графику видно, что данных об отправке весной у нас нет — это произошло, потому что после фильтрации оставалось 1 значение, но было удалено, т.к. был пропуск в целевой переменной. Так же заметно, что существенных отличий по медианному сроку доставки между сезонами не наблюдается. Однако для летнего периода характерна чуть большая медиана и более длинные "усы" боксплота, что может отражать влияние пиковых нагрузок или сезонных работ на инфраструктуре.

Для получения более полного представления о взаимосвязях между всеми количественными признаками была построена корреляционная матрица (рис. 8). Корреляционный анализ показал, что наиболее выраженная положительная связь с целевой переменной наблюдается у признаков «Нормативное количество дней доставки» и «Расстояние общее (км)», а также у веса груза, однако значения коэффициентов находятся в пределах слабой и средней силы связи. Прочие признаки имеют невысокие значения корреляции с целевой переменной, что говорит об отсутствии явных линейных зависимостей в данных. Данный результат подчёркивает необходимость использования сложных моделей, способных выявлять скрытые и нелинейные взаимосвязи между признаками.



*Рисунок 8. Корреляционная матрица*

В целом проведённый анализ позволяет сделать вывод о том, что сроки доставки зависят от комплекса факторов, среди которых значимую роль играют расстояние перевозки, нормативные параметры, а также возможные сезонные и маршрутные особенности.

## **4. Генерация признаков**

### **4.1 Создание новых признаков**

В первую очередь была создана агрегированная переменная «Код направления», объединяющая дорогу отправления и дорогу назначения в один уникальный код направления (рис. 15). Это позволяет учесть специфику конкретных логистических маршрутов.

*Листинг 7. Создание переменной «Код направления»*

dfv['Код направления'] = dfv['Дорога отправления'].astype(*str*) + "→" + dfv['Дорога назначения'].astype(*str*)

Далее для обогащения данных был реализован учёт праздников, что может отражаться на сроках доставки (листинг 8). Была сформирована коллекция нескольких праздничных дней, включая государственные и общеизвестные даты.

*Листинг 8. Создание переменной «Праздничный день»*

holidays = [  
 (1, 1), *# Новый год* (1, 7), *# Рождество* (2, 23), *# День защитника Отечества* (3, 8), *# Международный женский день* (5, 1), *# Праздник Весны и Труда* (5, 9), *# День Победы* (6, 12), *# День России* (11, 4), *# День народного единства* (12, 31), *# Предновогодний день (часто сокращённый/выходной)* (12, 25) *# Католическое Рождество (часто нерабочий в международных логистических компаниях)*]  
dfv['День'] = dfv['Дата начала'].dt.day  
dfv['Месяц'] = dfv['Дата начала'].dt.month  
dfv['Праздничный день'] = dfv[['Месяц', 'День']].apply(*lambda* x: (x['Месяц'], x['День']) *in* holidays, axis=1).astype(*int*)

На данном этапе формируется итоговая структура данных для дальнейшего моделирования, представленная таблицей 2, где в числовые признаки включён также новый признак «Праздничный день», а в категориальные — «Код направления».

*Таблица 2*

|  |  |
| --- | --- |
| **Тип признаков** | **Значения** |
| Числовые признаки | Вес груза (кг), Расстояние общее (км), Пробег общий (км), Нормативное количество дней доставки, Месяц отправки, День недели отправки, Выходной день, Время года, Праздничный день |
| Категориальные признаки | Дорога отправления, Дорога назначения, Государство отправления, Код направления |
| Целевая переменная | Срок доставки факт |
| Размер витрины | 6106,14 |

### **4.2 Кодирование категориальных переменных**

Для корректной работы моделей и возможности использовать категориальные признаки с большим числом уникальных значений применялось target encoding для признаков «Государство отправления», «Дорога отправления», «Дорога назначения», «Код направления» (листинг 9). Такой подход был выбран, поскольку в данных столбцах содержится много уникальных значений, и использование стандартного one-hot encoding привело бы к избыточному увеличению размерности признаков и появлению шума.

*Листинг 9. Кодирование категориальных переменных (Target Encoding)*

target\_cols = ['Государство отправления', 'Дорога отправления', 'Дорога назначения', 'Код направления']  
*for* col *in* target\_cols:  
 encoder = TargetEncoder(smoothing=10) *# Регуляризация* dfv[f'{col} (target)'] = encoder.fit\_transform(dfv[[col]], dfv['Срок доставки факт']).values.ravel()  
  
dfv.drop(columns=target\_cols, inplace=*True*)

Использование target encoding позволило превратить категориальные признаки в числовые, при этом сохранив потенциально полезную для модели информацию о влиянии конкретных категорий на целевую переменную. Столбец «Категория доставки\_По плану» остался референсным и не подвергался кодированию.

### **4.3 Устранение выбросов**

Для повышения устойчивости модели к аномальным значениям были проведены мероприятия по устранению выбросов, представленные листингом 10. В данном случае было принято решение использовать метод процентилей, поскольку исходное распределение целевой переменной заметно скошено и не приближается к нормальному.

*Листинг 10. Устранение выбросов в целевой переменной*

*#витрина для фильтрации*dffilt = dfv.copy()  
  
*#по целевой переменной (Срок доставки факт)*q01 = dffilt['Срок доставки факт'].quantile(0.01)  
q99 = dffilt['Срок доставки факт'].quantile(0.99)  
dffilt = dffilt[(dffilt['Срок доставки факт'] >= q01) & (dffilt['Срок доставки факт'] <= q99)]

## **5. Разделение на выборки**

Для корректной оценки качества построенной модели и предотвращения переобучения датасет был разделён на обучающую и тестовую выборки (листинг 11). Разделение выполнено по классическому стандарту — 80% данных выделено под обучение, оставшиеся 20% — для тестирования модели. В первую очередь из итоговой витрины были определены признаки (фичи) и целевая переменная. Далее данные были разбиты с помощью функции train\_test\_split из библиотеки scikit-learn.

*Листинг 11. Разделение на выборки*

*#фичи и таргет*target = 'Срок доставки факт'  
X = dffilt.drop(columns=[target])  
y = dffilt[target]  
  
*#разделяем на обучающую и тестовую выборки 80 на 20*X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42, shuffle=*True*)

В результате обучающая выборка содержит 4836 строк и 13 признаков, тестовая — 1210 строк и 13 признаков. Благодаря перемешиванию данных (shuffle=True) снижается риск получения перекошенных результатов из-за временных или структурных сдвигов в исходных данных.

## **6. Построение модели**

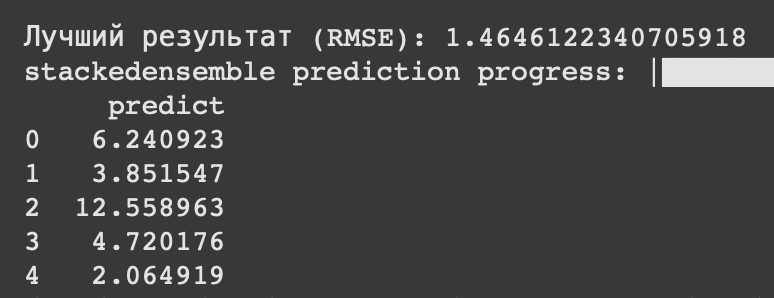
### **6.1 Baseline-модель с использованием AutoML**

Для получения базовой оценки качества прогноза была использована библиотека H2OAutoML. Вначале обучающая и тестовая выборки были преобразованы в формат H2OFrame, целевая переменная была названа «target» (листинг 12). Запуск AutoML производился с ограничением времени на 900 секунд и фиксированным seed для воспроизводимости результатов.

*Листинг 12. Построение baseline-модели (AutoML)*

train\_aml = h2o.H2OFrame(X\_train.assign(target=y\_train))  
test\_aml = h2o.H2OFrame(X\_test.assign(target=y\_test))  
  
features = X\_train.columns.tolist()  
target = 'target'  
  
aml = H2OAutoML(  
 seed=42,  
 verbosity="info",  
 max\_runtime\_secs=900  
)  
aml.train(x=features, y=target, training\_frame=train\_aml)

В результате работы AutoML лучшей оказалась ансамблевая модель типа StackedEnsemble (рис. 9).



*Рисунок 9. Результаты StackedEnsemble*

### **6.2 Моделирование**

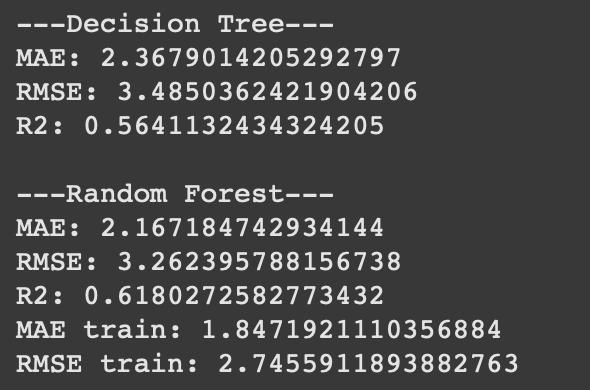
Для задачи прогнозирования срока доставки была реализована серия моделей, чтобы сравнить качество предсказания у разных типов алгоритмов и выбрать наиболее эффективный подход для данных, где важную роль играют как числовые, так и категориальные признаки.

На первом этапе были обучены простое дерево решений и ансамблевый метод — случайный лес (листинг 13). Это базовые алгоритмы для табличных данных, позволяющие построить интерпретируемые модели и оценить значимость признаков.

*Листинг 13. Модели на основе дерева решений и случайного леса*

*#DecisionTree*tree = DecisionTreeRegressor(max\_depth=6, random\_state=42)  
tree.fit(X\_train, y\_train)  
y\_pred\_tree = tree.predict(X\_test)  
  
*#RandomForest*forest = RandomForestRegressor(n\_estimators=100, max\_depth=8, random\_state=42)  
forest.fit(X\_train, y\_train)  
y\_pred\_forest = forest.predict(X\_test)  
*#предсказания на обучающей*y\_pred\_forest\_train = forest.predict(X\_train)

Модели показали значения метрик, представленные на рисунке 10. Несмотря на это, обе классические модели уступили по точности базовому ансамблю AutoML, который обеспечил RMSE = 1.46 на тех же данных.

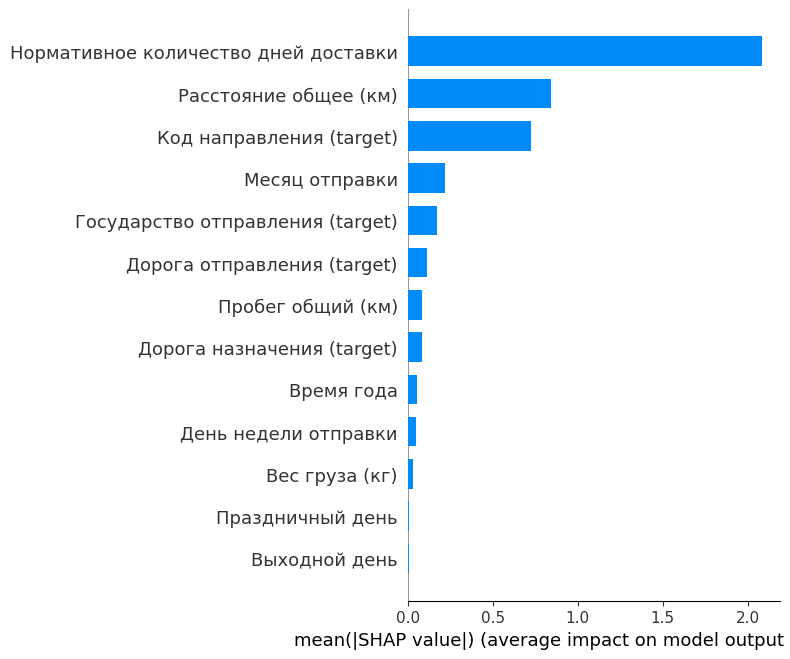


*Рисунок 10. Значения метрик построенных моделей*

Для случайного леса была рассчитана важность признаков. Наибольший вклад в предсказание срока доставки вносят:

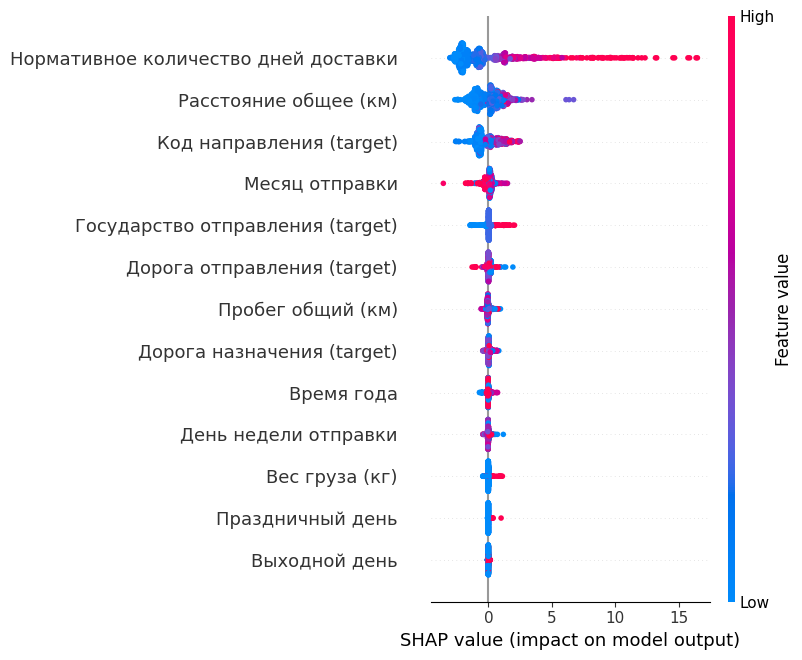
1. Нормативное количество дней доставки (feature importance: 0.591)
2. Расстояние общее (0.136)
3. Код направления (0.112)

Для дополнительной интерпретации использовался анализ SHAP-значений. На barplot-графике SHAP видно, что наиболее значимым признаком выступает нормативное количество дней доставки, за ним следуют расстояние и код направления (рис. 11).



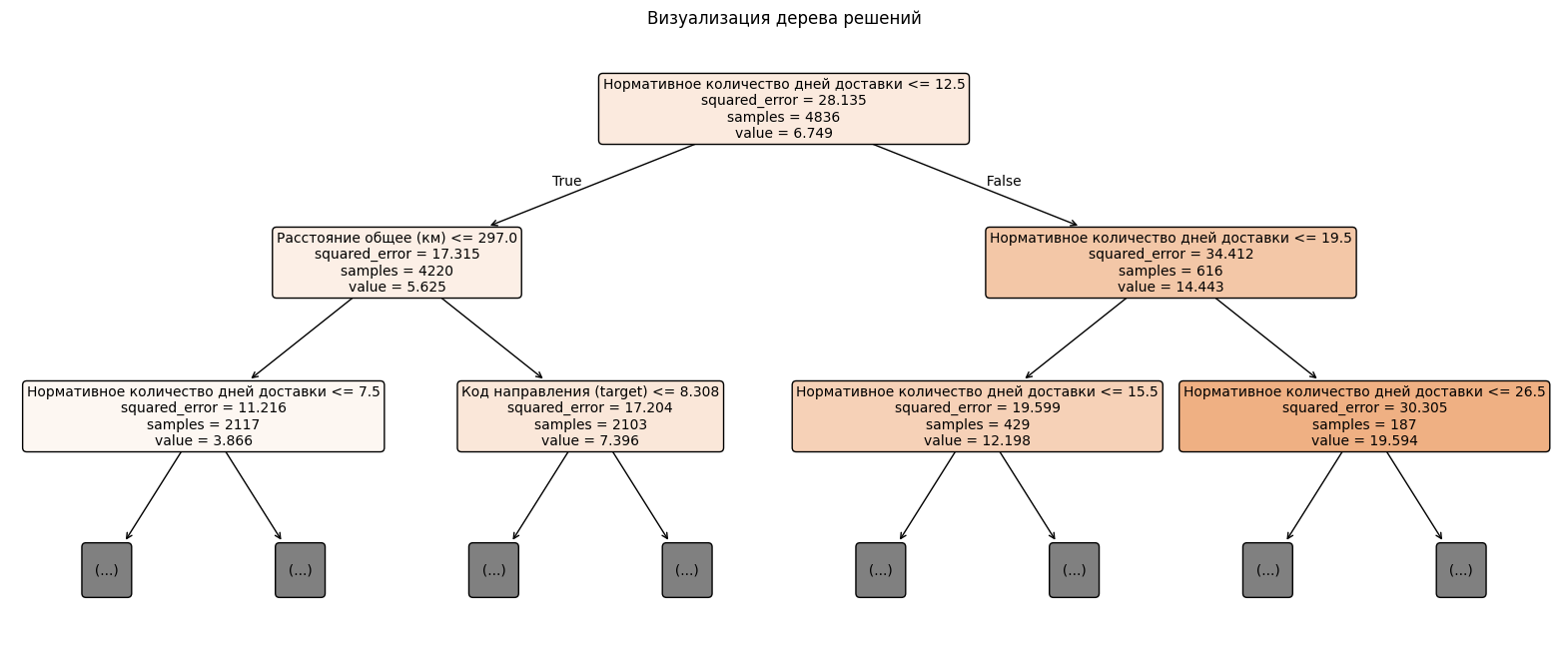
*Рисунок 11. Barplot-график SHAP*

SHAP summary plot показывает (рис. 12), что вклад этих признаков сохраняется стабильным как в отдельных наблюдениях, так и на общем уровне.



*Рисунок 12. SHAP summary plot*

Для простоты объяснения структуры модели была выполнена визуализация построенного дерева решений (рис. 13). Из структуры дерева видно, что на первых разбиениях оказывают влияние те же признаки, что были выявлены как ключевые по важности: нормативное количество дней доставки, расстояние общее и код направления.



*Рисунок 13. Визуализация дерева решения*

Далее были добавлены современные градиентные бустинги (CatBoost, XGBoost, GradientBoosting), а также линейные модели Ridge и Lasso, и KNeighborsRegressor как baseline для instance-based подхода (листинг 14). Это позволило проверить, насколько сложные ансамбли превосходят простые модели на этих данных.

*Листинг 14. Градиентные бустинги и линейная модель*

*#1 CatBoost  
from* catboost *import* CatBoostRegressor  
cat = CatBoostRegressor(iterations=300, learning\_rate=0.1, depth=6, random\_seed=42, verbose=*False*)  
cat.fit(X\_train, y\_train)  
y\_pred\_cat\_train = cat.predict(X\_train)  
y\_pred\_cat = cat.predict(X\_test)  
print\_metrics(y\_train, y\_pred\_cat\_train, y\_test, y\_pred\_cat, "CatBoost")

*# XGBoost  
from* xgboost *import* XGBRegressor  
xgb = XGBRegressor(n\_estimators=300, learning\_rate=0.1, max\_depth=6, random\_state=42, verbosity=0)  
xgb.fit(X\_train, y\_train)  
y\_pred\_xgb\_train = xgb.predict(X\_train)  
y\_pred\_xgb = xgb.predict(X\_test)  
print\_metrics(y\_train, y\_pred\_xgb\_train, y\_test, y\_pred\_xgb, "XGBoost")  
  
*# GradientBoostingRegressor  
from* sklearn.ensemble *import* GradientBoostingRegressor  
gbr = GradientBoostingRegressor(n\_estimators=300, learning\_rate=0.1, max\_depth=6, random\_state=42)  
gbr.fit(X\_train, y\_train)  
y\_pred\_gbr\_train = gbr.predict(X\_train)  
y\_pred\_gbr = gbr.predict(X\_test)  
print\_metrics(y\_train, y\_pred\_gbr\_train, y\_test, y\_pred\_gbr, "GradientBoostingRegressor")  
  
*# Ridge  
from* sklearn.linear\_model *import* Ridge  
ridge = Ridge(alpha=1.0)  
ridge.fit(X\_train, y\_train)  
y\_pred\_ridge\_train = ridge.predict(X\_train)  
y\_pred\_ridge = ridge.predict(X\_test)  
print\_metrics(y\_train, y\_pred\_ridge\_train, y\_test, y\_pred\_ridge, "Ridge")

Результаты:

* CatBoost: MAE = 2.15, RMSE = 3.21, R2 = 0.63
* XGBoost: MAE = 2.11, RMSE = 3.22, R2 = 0.63
* GradientBoostingRegressor: MAE = 2.11, RMSE = 3.21, R2 = 0.63
* Ridge: MAE = 2.57, RMSE = 3.72, R2 = 0.50

Итого, современные ансамблевые методы на базе градиентного бустинга (CatBoost, XGBoost, GradientBoosting) показали самые высокие результаты по всем метрикам на тестовой выборке, опередив даже случайный лес. Линейная модель сильно уступают — это говорит о том, что в задаче присутствуют сложные, нелинейные зависимости, которые хорошо улавливают только продвинутые ансамбли.

## **7. Лучшая модель**

Лучшие результаты были достигнуты с помощью ансамбля моделей, автоматически построенного через AutoML. StackedEnsemble занял первое место по всем основным метрикам:

* RMSE: **1.45**
* MAE: **1.97**

Ансамбль AutoML (StackedEnsemble) показал минимальные значения ошибок. Такой результат объясняется тем, что стекинг объединяет преимущества различных алгоритмов (градиентные бустинги, линейные модели, деревья решений), а AutoML автоматически подбирает оптимальные параметры и проводит кросс-валидацию. Это обеспечивает наибольшую устойчивость и точность прогноза.

**Современные бустинговые модели (CatBoost, XGBoost, GradientBoosting):**  
В ручном сравнении градиентные бустинги показали схожий высокий уровень качества:

* **CatBoost**: RMSE = 3.21, MAE = 2.15, R² = 0.63
* **XGBoost**: RMSE = 3.22, MAE = 2.11, R² = 0.63
* **GradientBoostingRegressor**: RMSE = 3.21, MAE = 2.11, R² = 0.63

Эти модели превосходят простое дерево решений и линейный метод, подтверждая, что задача требует сложных, нелинейных зависимостей и учёта взаимодействий между признаками.

**Random Forest Regressor:** Случайный лес дал лучший результат среди классических «ручных» ансамблей:

* RMSE: **3.26**
* MAE: **2.17**
* R²: **0.62**

Random Forest уступает по точности как AutoML, так и градиентным бустингам, но сохраняет хорошую интерпретируемость и устойчивость к переобучению.

**Линейная модель (Ridge).** Ошибки этой модели выше. Это наглядно показывает, что линейные методы не способны уловить сложные закономерности в этих данных.

**Decision Tree Regressor.** Дерево решений использовалось для интерпретации, а также как простая baseline-модель:

* RMSE: **3.49**
* MAE: **2.37**
* R²: **0.56**

Дерево решений предсказывает заметно хуже, склонно к переобучению и служит скорее инструментом визуализации логики решений, чем средством получения высокой точности.

Лучшая модель — это **ансамбль StackedEnsemble из AutoML**, показавший наименьшую ошибку прогноза за счёт сочетания сильных сторон разных алгоритмов и автоматизации отбора параметров. Среди ручных моделей лидируют современные градиентные бустинги, а Random Forest выступает как надёжная альтернатива с высокой интерпретируемостью.  
Простые линейные модели и KNN не подходят для данной задачи из-за ограничений на работу с нелинейными зависимостями.

## **8. Интерпретация лучшей модели**

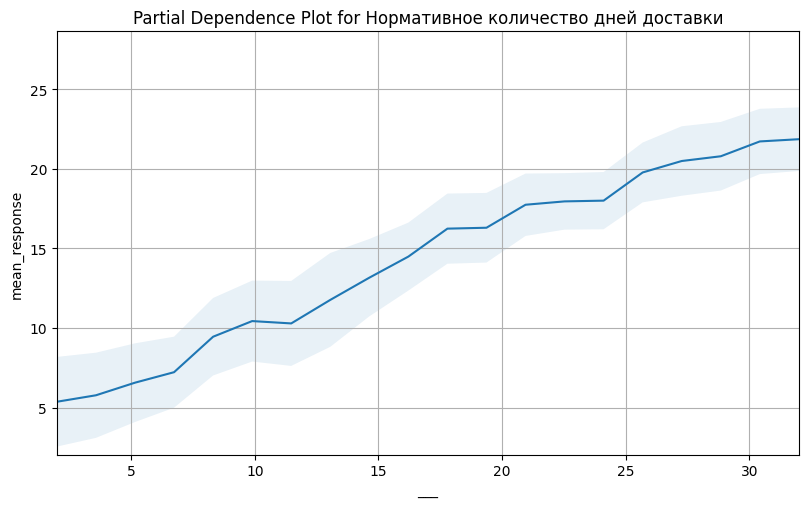
### **8.1 Глобальная интерпретация**

Глобальная интерпретация лучшей модели (AutoML StackedEnsemble) проводится с использованием partial dependence plots (PDP) — графиков частичной зависимости, которые позволяют увидеть, как конкретный признак влияет на итоговый прогноз при прочих равных условиях. Построение PDP для двух ключевых признаков представлено листингом 14.

*Листинг 14. Partial dependence plots*

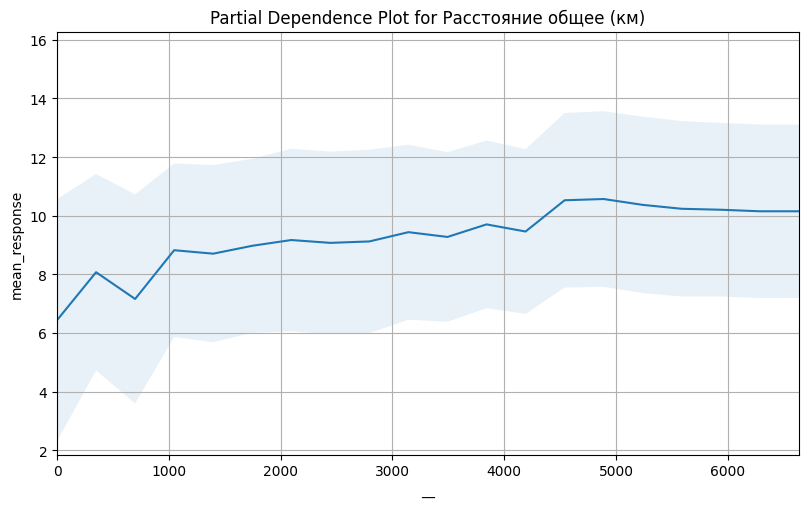
aml.leader.partial\_plot(  
 data=test\_aml,  
 cols=['Нормативное количество дней доставки'],  
 plot=*True*,  
 nbins=20,  
 figsize=(8, 5)  
)  
aml.leader.partial\_plot(  
 data=test\_aml,  
 cols=['Расстояние общее (км)'],  
 plot=*True*,  
 nbins=20,  
 figsize=(8, 5)  
)

Первый график (рис. 14) показывает ярко выраженную возрастающую зависимость: чем больше нормативный срок, тем больше модель прогнозирует фактический срок доставки. Это абсолютно логично: если по нормативу на перевозку выделяется больше дней, и модель ожидает, что и факт тоже будет больше. Влияние признака почти линейное и однозначное, что подтверждает его ключевую роль.



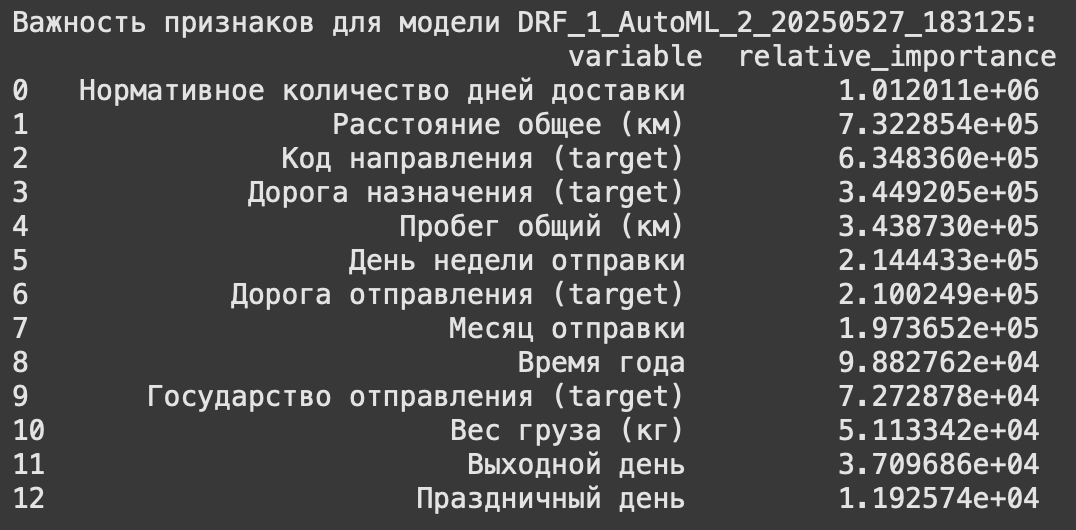
*Рисунок 14. Первый график PDP*

Второй график (рис. 15) отражает умеренно возрастающую зависимость между расстоянием перевозки и прогнозируемым сроком. При малых расстояниях (до ~1000 км) прогнозируемый срок доставки — около 6–8 дней. При увеличении расстояния до 5000–6000 км прогноз увеличивается до 10–11 дней, а далее рост замедляется или даже снижается. Широкий доверительный интервал для больших расстояний говорит о меньшей уверенности модели в этих областях данных.



*Рисунок 15. Второй график PDP*

Далее была рассчитана важность признаков для StackedEnsemble. Поскольку для StackedEnsemble напрямую важность признаков недоступна, для анализа был взят следующий по качеству алгоритм (DRF из AutoML), где возможен корректный анализ importance. Результаты отображены на рисунке 16.



*Рисунок 16. Важность признаков*

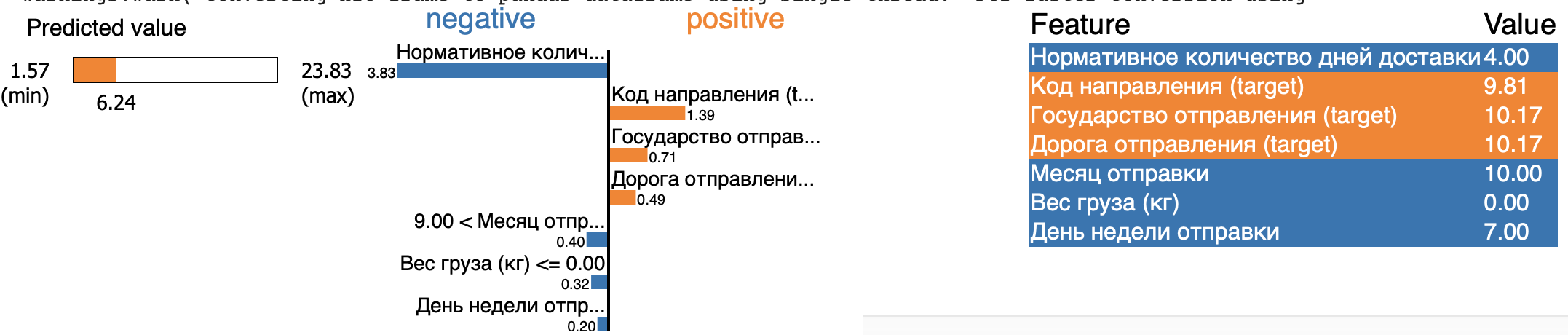
### **8.2 Локальная интерпретация**

Для локальной интерпретации прогноза лучшей модели был использован метод LIME (Local Interpretable Model-agnostic Explanations). Это позволило подробно разобрать вклад каждого признака для отдельных прогнозов.

*Листинг 15. Метод LIME*

idxs = [0, 1, 2]  
  
*#для лайма нужна predict\_fn, которая возвращает нампай-массив  
def* h2o\_predict\_fn(X):  
 *import* pandas *as* pd  
 X\_df = pd.DataFrame(X, columns=X\_test.columns)  
 h2o\_df = h2o.H2OFrame(X\_df)  
 preds = aml.leader.predict(h2o\_df).as\_data\_frame().values.ravel()  
 *return* preds  
  
*#инициализация*explainer = lime.lime\_tabular.LimeTabularExplainer(  
 training\_data=np.array(X\_train),  
 feature\_names=X\_train.columns.tolist(),  
 mode='regression'  
)  
  
*#объясняем прогнозы для выбранных наблюдений  
for* idx *in* idxs:  
 exp = explainer.explain\_instance(  
 np.array(X\_test.iloc[idx]),  
 h2o\_predict\_fn,  
 num\_features=7  
 )  
 *print*(f"\n----------Интерпретация для наблюдения {idx}-----------")  
 exp.show\_in\_notebook(show\_table=*True*)

Для каждого конкретного прогноза будет наглядно видно, какие именно признаки двигали результат вверх или вниз. Например, для первого наблюдения (рис. 17) основные факторы, увеличивающие прогноз, — это значения по коду направления, государству отправления и дороге отправления, в то время как нормативный срок оказывает обратный эффект и снижает итоговый прогноз.



*Рисунок 17. Результаты LIME для первого наблюдения*

## **9. Дополнительные решения по работе**

### **9.1 Permutation Feature Importance**

Метод оценки вклада признаков основан на идее: если "перемешать" значения одного признака по выборке и пересчитать ошибку модели, можно понять, насколько этот признак важен для прогноза. Чем сильнее увеличивается ошибка — тем значимее признак (листинг 16).

*Листинг 16. Перестановочная важность признаков*

result = permutation\_importance(forest, X\_test, y\_test, n\_repeats=10, random\_state=42)  
  
*print*("Permutation Feature Importance:")  
*for* i *in* result.importances\_mean.argsort()[::-1]:  
 *print*(f"{X\_test.columns[i]}: {result.importances\_mean[i]:.3f}")

Результаты Permutation Feature Importance выявили топ признаков:

1. Нормативное количество дней доставки: 1.013
2. Расстояние общее (км): 0.209
3. Код направления (target): 0.106
4. Месяц отправки: 0.043
5. Государство отправления (target): 0.022
6. Дорога отправления (target): 0.020
7. Дорога назначения (target): 0.013
8. Время года: 0.009

Результаты подтверждают предыдущее распределение важности: главные признаки — нормативный срок, расстояние и код направления.

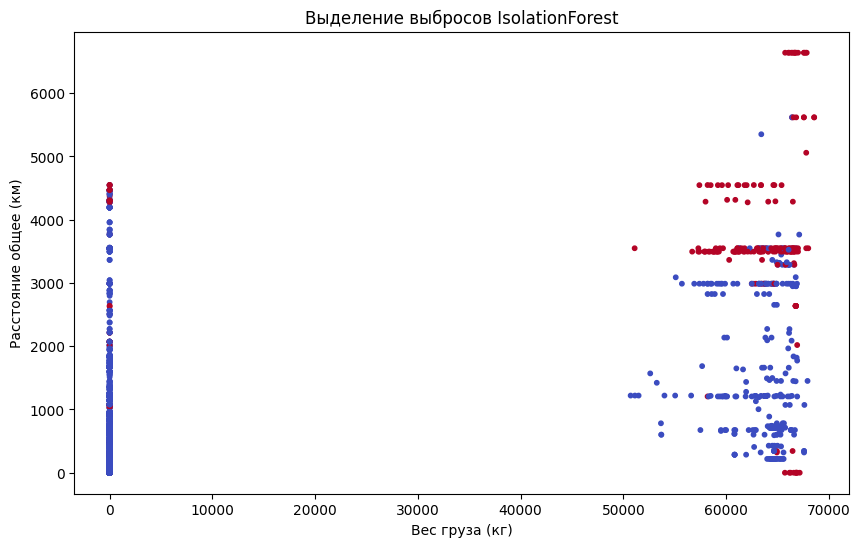
### **9.2 Outlier Detection — Isolation Forest**

Для анализа качества и структуры данных дополнительно использовался алгоритм Isolation Forest для выделения выбросов (аномальных наблюдений), которые могут негативно влиять на обучение модели (листинг 17).

*Листинг 17. Алгоритм Isolation Forest*

iso = IsolationForest(contamination=0.05, random\_state=42)  
outlier\_labels = iso.fit\_predict(X\_train)  
print("Количество выбросов:", (outlier\_labels==-1).sum())  
  
plt.scatter(X\_train.iloc[:,0], X\_train.iloc[:,1], c=(outlier\_labels==-1), cmap='coolwarm', s=10)  
plt.xlabel(X\_train.columns[0])  
plt.ylabel(X\_train.columns[1])  
plt.title("Выделение выбросов IsolationForest")  
plt.show()

Количество выбросов: 242. Аномалии в данных были выявлены по сочетаниям «Вес груза (кг)» и «Расстояние общее (км)». Хотя вес не является ключевым для прогноза, именно он хорошо выделяет отдельные аномальные группы (например, записи с очень тяжёлыми грузами или редкими сочетаниями большого веса и длинного расстояния). На графике видно (рис. 18), что большинство выбросов — это либо очень тяжёлые грузы, либо редкие сочетания значений.



*Рисунок 18. Выделение выбросов Isolation Forest*

### **9.3 Dummy Regressor (базовая модель для сравнения)**

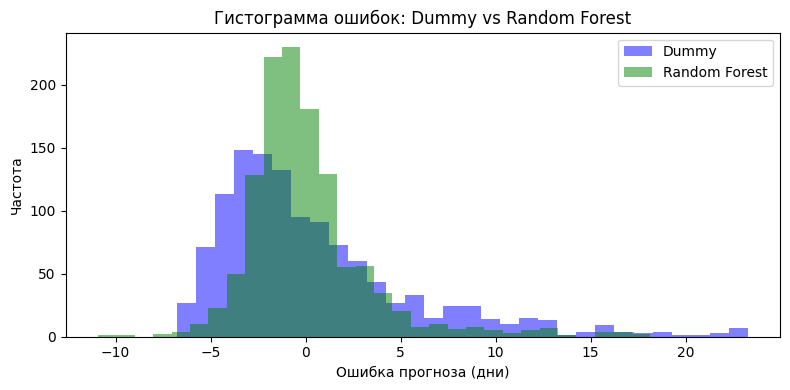
Для корректной оценки качества любых умных моделей обязательно требуется сравниваться с простым бэйзлайном — DummyRegressor (стратегия «mean»). Это простая модель, которая всегда прогнозирует среднее по обучающей выборке (листинг 18).

*Листинг 18. Алгоритм DummyRegressor*

dummy = DummyRegressor(strategy="mean")  
dummy.fit(X\_train, y\_train)  
y\_pred\_dummy = dummy.predict(X\_test)  
*print*("Dummy RMSE:", np.sqrt(mean\_squared\_error(y\_test, y\_pred\_dummy)))

Dummy RMSE: 5.28. Результат DummyRegressor намного хуже, чем у AutoML и Random Forest (там ошибки примерно в 2–3 раза меньше). Это говорит о том, что построенные модели действительно учат закономерности, а не просто угадывают среднее.

Для дополнительной визуализации был построен график распределения ошибок Dummy и Random Forest, представленный на рисунке 19.



*Рисунок 19. Гистограмма ошибок*

Вывод по графику: Random Forest даёт меньше ошибок, распределение ошибок более "узкое"; количество крупных ошибок у Random Forest существенно меньше, в отличие от Dummy он практически не допускает сильных промахов.

### **9.4 Собственная метрика качества: MAE по праздничным дням**

Для бизнес-задач особенно важно, чтобы модель была устойчива в критических ситуациях — например, в праздники, когда сроки меняются из-за особенностей расписания. MAE по праздничным дням — средняя абсолютная ошибка только по тем примерам, где дата пришлась на праздник (листинг 19).

*Листинг 19. MAE по праздничным дням*

holiday\_idx = X\_test['Праздничный день'] == 1  
mae\_holiday = mean\_absolute\_error(y\_test[holiday\_idx], y\_pred\_forest[holiday\_idx])  
*print*('MAE по праздничным дням:', mae\_holiday)

MAE по праздничным дням: 2.07 (для сравнения: общий MAE для StackedEnsemble = 1.97). Зачем нужна эта метрика? В логистике праздники — это всегда особые условия: меняется график, часто есть ограничения по срокам и увеличиваются риски опозданий. Ошибка на праздниках критичнее, чем в обычные дни. Поэтому если MAE по праздникам низкое — модель хорошо справляется с такими "тяжёлыми" случаями, если высокое — бизнес рискует не выполнить обязательства в сложные дни, но у нас МАЕ примерно одинаковые, значит модель хорошо справляется.

# **Выводы**

В рамках курсовой работы была решена практическая задача построения и интерпретации модели машинного обучения для прогноза сроков доставки грузов по железнодорожной логистике. Работа прошла все ключевые этапы: от анализа и очистки данных до построения, сравнения и объяснения нескольких моделей, а также проверки устойчивости к выбросам и особым бизнес-ситуациям (праздничные дни).

1. Анализ и подготовка данных.

Проведён подробный первичный анализ, выявлены особенности структуры данных: множество категориальных и временных признаков, существенные пропуски и наличие выбросов. Были реализованы методы очистки, фильтрации, генерации новых признаков и нормализации временных столбцов. Особое внимание уделено смысловой подготовке данных — преобразованию дат, генерации бинарных и сезонных признаков, а также корректному отбору целевой переменной.

2. Исследование признаков.

Проведён анализ распределений, корреляций и пропусков, построены визуализации по ключевым характеристикам. Выявлено, что наибольший вклад в целевую переменную оказывают нормативный срок доставки и расстояние перевозки — что подтверждается как корреляционным, так и более сложным анализом (feature importance, PDP, SHAP).

3. Моделирование и сравнение алгоритмов.

Для прогнозирования сроков доставки протестированы несколько моделей: Decision Tree, Random Forest и ансамблевый AutoML (StackedEnsemble). Наилучший результат по метрикам качества (RMSE и MAE) показал ансамбль AutoML, что ожидаемо — стеккинг позволяет объединять преимущества разных методов. Random Forest стал лучшим среди «ручных» моделей и подтвердил информативность выбранных признаков.

4. Интерпретация и объяснения модели.

Для понимания логики работы финальной модели проведён как глобальный, так и локальный анализ:

* Использованы Partial Dependence Plots, permutation importance и SHAP-значения для глобального анализа вкладов признаков.
* Метод LIME позволил разобрать, как модель принимает решения для конкретных индивидуальных случаев.

5. Проверка устойчивости, альтернативные метрики и анализ ошибок.

* В работе дополнительно оценены:
* Устойчивость моделей к выбросам (через Isolation Forest),
* Качество по бизнес-критичной метрике (MAE по праздничным дням),
* Сравнение с DummyRegressor, что подтвердило «неслучайность» построенных моделей.
* Проведён анализ распределения ошибок, показавший стабильную работу модели даже в сложных сегментах данных.

Общий итог: построена интерпретируемая и устойчивая модель прогнозирования сроков доставки, обладающая высокой точностью и стабильностью в реальных и сложных ситуациях. Применён широкий спектр современных методов анализа данных и интерпретации моделей, что позволяет объяснять полученные прогнозы как на уровне выборки, так и для отдельных наблюдений. Все полученные результаты подтверждаются альтернативными экспериментами и дополнительными метриками, а значит — выводы и качество модели надёжны.

Практическая ценность: результаты работы могут быть использованы для оптимизации процессов планирования перевозок, прогнозирования рисков и повышения удовлетворённости клиентов за счёт более точных сроков поставки.