Оглавление

Введение

В настоящее время методы классификации в области обработки больших данных находят применения в множестве сфер. Одним из примеров можно привести применение классификаци в банковской системе: там она помогает в определение кредитоспособности заёмщиков. Кроме того, систему подобного рода можно настроить на решение других задач классификации, при условии, что обучающая выборка и данные будут достаточно репрезентативны и представлять из себя статистически значимые признаки.

Это, в свою очередь, предоставляет большие возможности для развития подобных систем в различных сферах повседневной жизни. Поэтому тема представляет большой интерес с точки зрения исследования.

Цель работы

Иссследовать методы классификации на основе предоставленного датасета **flsr_moscow.txt**, разработать программу для классификации квартир на четыре класса по параметру площади (**totsp**) и сравнить полученные результаты для распределения квартир по количеству комнат.

Задачи

— Изучить теоретические основы методов классификации
— Проанализировать полученный датасет
— Построить модель классификации типа softmax
— Провалидировать полученную модель
— Построить матрицу путаницы
— Провести ROC-анализ
— Сравнить методы softmax классификации и SVM
— Визуализировать полученные результаты

1 Теоретическая часть

1 Математическое описание модели `multinom`

Модель `multinom` основана на многочленной логистической регрессии, которая используется для предсказания категориальной зависимой переменной с K классами ($K > 2$) на основе вектора признаков $\mathbf{x} \in \mathbb{R}^p$.

1.1 Вероятности классов (softmax)

Вероятность принадлежности наблюдения i-му классу k задаётся следующим выражением:

$$P(y_i = k \mid \mathbf{x}_i) = \frac{\exp(\beta_{k0} + \beta_k^\top \mathbf{x}_i)}{1 + \sum_{j=1}^{K-1} \exp(\beta_{j0} + \beta_j^\top \mathbf{x}_i)}, \quad k = 1, \ldots, K-1.$$

Для последнего класса ($k = K$) используется нормализация:

$$P(y_i = K \mid \mathbf{x}_i) = \frac{1}{1 + \sum_{j=1}^{K-1} \exp(\beta_{j0} + \beta_j^\top \mathbf{x}_i)}.$$

Здесь:

— β_{k0} — свободный член для класса k;

— $\beta_k = (\beta_{k1}, \beta_{k2}, \ldots, \beta_{kp})^\top \in \mathbb{R}^p$ — вектор коэффициентов для класса k.

1.2 Функция правдоподобия

Параметры модели оцениваются с помощью максимизации логарифма функции правдоподобия:

$$\ell(\boldsymbol{\beta}) = \sum_{i=1}^{n} \sum_{k=1}^{K} \mathbb{1}(y_i = k) \ln P(y_i = k \mid \mathbf{x}_i),$$

где:

— $\mathbb{1}(y_i = k)$ — индикатор принадлежности наблюдения i-му классу k, равный 1, если $y_i = k$, и 0 в противном случае;

— $P(y_i = k \mid \mathbf{x}_i)$ задаётся формулой вероятности классов (см. выше).

1.3 Процедура оптимизации

Параметры $\{\beta_{k0}, \beta_k\}_{k=1}^{K-1}$ вычисляются с использованием численных методов оптимизации. Наиболее распространённые методы:

— метод Ньютона-Рафсона, который использует градиенты и матрицу Гессе;

— итеративные методы максимального правдоподобия.

Градиент функции логарифмического правдоподобия по $\boldsymbol{\beta}$ вычисляется как:

$$\nabla_{\boldsymbol{\beta}} \ell(\boldsymbol{\beta}) = \sum_{i=1}^{n} \sum_{k=1}^{K} \mathbb{1}(y_i = k) \left[\frac{\partial \ln P(y_i = k \mid \mathbf{x}_i)}{\partial \boldsymbol{\beta}} \right].$$

1.4 Предсказание значений

Для оценки класса объекта по обученной модели используется следующий подход:

а) Вычисляются вероятности принадлежности объекта каждому из K классов, используя формулы:

$$\hat{P}(y = k \mid \mathbf{x}) = \frac{\exp(\hat{\beta}_{k0} + \hat{\beta}_k^\top \mathbf{x})}{1 + \sum_{j=1}^{K-1} \exp(\hat{\beta}_{j0} + \hat{\beta}_j^\top \mathbf{x})}, \quad k = 1, \ldots, K-1,$$

и для последнего класса ($k = K$):

$$\hat{P}(y = K \mid \mathbf{x}) = \frac{1}{1 + \sum_{j=1}^{K-1} \exp(\hat{\beta}_{j0} + \hat{\beta}_j^\top \mathbf{x})}.$$

б) Определяется класс с максимальной вероятностью:

$$\hat{y} = \arg\max_k \hat{P}(y = k \mid \mathbf{x}).$$

1.5 Оценка модели

Оценка модели проводится с использованием различных метрик, включая матрицу путанности (confusion matrix) и анализ ROC-кривых.

1.5.1 Матрица путанности

Матрица путанности показывает количество объектов, предсказанных и реальных, для каждого класса. Пример структуры матрицы для K-классовой задачи:

$$\text{Confusion Matrix} = \begin{bmatrix} \text{TP}_{11} & \text{FP}_{12} & \cdots & \text{FP}_{1K} \\ \text{FP}_{21} & \text{TP}_{22} & \cdots & \text{FP}_{2K} \\ \vdots & \vdots & \ddots & \vdots \\ \text{FP}_{K1} & \text{FP}_{K2} & \cdots & \text{TP}_{KK} \end{bmatrix},$$

где:

— TP_{kk} — количество правильных предсказаний для класса k (true positives).

— FP_{ij} — количество объектов класса i, ошибочно предсказанных как класс j (false positives).

Метрику точности (*accuracy*), вычисляемую как:

$$\text{Accuracy} = \frac{\text{Количество правильных предсказаний (диагональные элементы)}}{\text{Общее количество объектов}}.$$

$$\text{Accuracy} = \frac{\sum_k TP_{kk}}{\sum_k TP_{kk} + \sum_{j,j,i\neq j} FP_{ij}}.$$

1.5.2 ROC-кривые

ROC (Receiver Operating Characteristic) кривые строятся для каждого класса k отдельно:

а) Рассчитывается вероятность $P(y = k \mid \mathbf{x})$ для всех объектов.

б) Выбираются различные пороги вероятности t, чтобы предсказывать принадлежность к классу k (например, если $P(y = k \mid \mathbf{x}) \geqslant t$, то предсказывается k).

в) Для каждого порога вычисляются:

— Доля истинно положительных результатов (TPR):

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}.$$

— Доля ложных положительных результатов (FPR):

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}.$$

г) На графике (FPR, TPR) строится ROC-кривая.

Площадь под ROC-кривой (*Area Under Curve*, AUC) используется для количественной оценки качества модели:

$$\text{AUC} \in [0, 1].$$

Чем ближе значение AUC к 1, тем лучше работает модель для классификации данного класса.

1.6 Дополнение

Для многоклассовых моделей оценивают средние значения метрик (например, ROC-AUC):

$$\text{ROC-AUC}_{\text{macro}} = \frac{1}{K} \sum_{k=1}^{K} \text{AUC}_k.$$

2 Модель `ksvm` с ядром `rbfdot`

Модель `ksvm` в пакете `kernlab` реализует метод опорных векторов (Support Vector Machine, SVM) для задач классификации или регрессии. Использование ядра `rbfdot` означает применение радиально-базисной функции (RBF kernel), которая позволяет обрабатывать нелинейно разделимые данные.

2.1 Математическое описание

Пусть у нас есть обучающая выборка $\{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$, где $\mathbf{x}_i \in \mathbb{R}^p$ — вектор признаков, а $y_i \in \{-1, +1\}$ для бинарной классификации или $y_i \in \{1, 2, \ldots, K\}$ для многоклассовой задачи.

Модель SVM решает следующую оптимизационную задачу:

$$\min_{\mathbf{w}, b, \xi} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{n} \xi_i,$$

при ограничениях:

$$y_i \left(\mathbf{w}^\top \phi(\mathbf{x}_i) + b \right) \geqslant 1 - \xi_i, \quad \xi_i \geqslant 0, \quad i = 1, \ldots, n.$$

Здесь:

— $\mathbf{w}$ — весовой вектор модели;

— b — смещение (bias);

— ξ_i — штрафы за ошибочную классификацию (slack variables);

— $C > 0$ — гиперпараметр, регулирующий степень регуляризации (баланс между максимизацией отступа и минимизацией ошибок классификации);

— $\phi(\mathbf{x}_i)$ — отображение в пространство большей размерности, определяемое ядром;

— $\|\mathbf{w}\|^2$ — квадрат нормы весов, который минимизируется для максимизации отступа.

2.2 RBF-ядро

Использование RBF-ядра (радиально-базисной функции) `rbfdot` определяет $\phi(\mathbf{x}_i)$ косвенно через ядровую функцию:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2\right),$$

где $\gamma > 0$ — параметр ядра, управляющий степенью воздействия расстояния между объектами на значения ядра.

RBF-ядро обладает высокой гибкостью, позволяя эффективно классифицировать данные с нелинейными границами.

2.3 Вывод вероятностей

Функция `ksvm` позволяет включить вероятностное моделирование (`prob.model = TRUE`). Это достигается через калибровку выходов модели методом Платта:

$$P(y = +1 \mid \mathbf{x}) = \frac{1}{1 + \exp(Af(\mathbf{x}) + B)},$$

где:

— $f(\mathbf{x}) = \mathbf{w}^\top \phi(\mathbf{x}) + b$ — дискриминантная функция SVM;

— A и B — параметры, оцениваемые методом минимизации логарифмической потери на обучающей выборке.

2.4 Оценка модели

Для оценки качества модели SVM используют:

— Метрику точности (*accuracy*), вычисляемую как:

$$\text{Accuracy} = \frac{\text{Количество правильных предсказаний}}{\text{Общее количество объектов}}.$$

— Матрицу путанности для анализа ошибок.

— ROC-кривые и AUC для вероятностных предсказаний (при `prob.model = TRUE`).

— Среднюю кросс-энтропию, если модель калибрует вероятности:

$$\text{Log Loss} = -\frac{1}{n}\sum_{i=1}^{n} \ln P(y_i \mid \mathbf{x}_i).$$

2 Программная часть

1 Подготовка данных

Листинг 2.1 — Импорт необходимых библиотек

```
1 library(nnet)
2 library(ggplot2)
3 library(dplyr)
4 library(caret)
5 library(pROC)
```

— **nnet**: Библиотека для создания нейронных сетей. Основные функции: `nnet()` для построения многослойных перцептронов и обучение моделей с использованием обратного распространения ошибки.

— **ggplot2**: Библиотека для создания графиков и визуализации данных. Поддерживает создание различных типов графиков на основе грамматики графиков.

— **dplyr**: Библиотека для манипуляции данными. Обеспечивает функции для фильтрации, сортировки, группировки и агрегации данных, упрощая работу с дата-фреймами.

— **caret**: Библиотека для машинного обучения. Предоставляет единый интерфейс для создания, оценки и настройки моделей, а также функции для предобработки данных и оценки производительности.

— **pROC**: Библиотека для анализа ROC-кривых. Позволяет вычислять и визуализировать ROC-кривые и площадь под кривой (AUC), упрощая оценку качества классификаторов.

Листинг 2.2 — Загрузка датасета и задание сида

```
1 data <- read.table(file="flsr_moscow.txt", header = TRUE)
2 head(data)
3 summary(data)
4 dim(data)
5 print(colSums(sapply(data, is.na)))
6
7 set.seed(42)
```

Данный код загружает датасет **flsr_moscow.txt** С учетом заголовков. С помощью метода **head()**, просматриваются первые 6 строк данных:

Таблица 1 — Данные объектов недвижимости

#	price	nrooms	totsp	livesp	kitsp	walk	...	code
1	81	1	58	40	6	1	...	3
2	75	1	44	28	6	1	...	6
3	95	1	61	37	6	1	...	1
4	98	1	59	39	6	0	...	8
5	88	1	55	36	6	1	...	4
6	96	1	60	43	6	1	...	7

Также формируется общая информация о датасете с помощью метода **summary()**:

Таблица 2 — Сводная статистика объектов недвижимости

Параметр	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
price	50.0	95.0	115.0	127.5	142.0	730.0
nrooms	1.000	1.000	2.500	2.491	3.000	4.000
totsp	44.00	62.00	73.50	73.08	79.00	192.00
livesp	28.00	42.00	45.00	46.34	50.00	102.00
kitsp	5.000	7.000	9.000	8.899	10.000	25.000
dist	3.00	9.00	12.00	11.02	13.50	17.00
metrdist	1.000	5.000	7.000	8.117	10.000	20.000
walk	0.0000	0.0000	1.0000	0.6858	1.0000	1.0000
brick	0.000	0.000	0.000	0.323	1.000	1.000
floor	0.0000	1.0000	1.0000	0.7907	1.0000	1.0000
code	1.000	3.000	4.000	4.322	6.000	8.000

Проверка размерности датасета (**dim()**): 2040 строк и 11 столбцов.

Проверка на полноту датасета (**print(colSums(sapply(data, is.na)))**) - сумма na значений по столбцам:

Таблица 3 — Данные объектов недвижимости (нулевые значения)

price	nrooms	totsp	livesp	kitsp	dist	metrdist	walk	brick	floor	code
0	0	0	0	0	0	0	0	0	0	0

Также, зададим сид веротности, равный 42.

Листинг 2.3 — Формирование классов на основании переменной totsp

```
data$class <- cut(data$totsp, breaks = 4, labels = c("small", "medium",
    "large", "huge"))

data <- select(data, -totsp)
dim(data)
summary(data$class)
```

Данный код добавляет новое поле **class** в датасет. Тип - факторный с четырьмя уровнями.

Механизм деления: метод **cut** делит целевой набор (totsp) на 4 части (breaks) с наименованиями **labels**.

Далее, удаляется переменная **totsp**. Проверяется размерность датасета: 2040 строк и 11 столбоцов. Ничего не было потеряно: один столбец убран и новый добавился.

Также выведем информацию по новому параметру:

Таблица 4 — Частота уровней факторной переменной

Уровень	Частота
small	1672
medium	329
large	36
huge	3

Листинг 2.4 — Формирвание обучающей, валидационной и тестовой выборок

```
train_index <- caret::createDataPartition(data$class, p = 0.6, list = FALSE)
train_data <- data[train_index, ]
temp_data <- data[-train_index, ]

validation_index <- caret::createDataPartition(temp_data$class, p = 0.5, list = FALSE)
validation_data <- temp_data[validation_index, ]
test_data <- temp_data[-validation_index, ]

cat("Size of train data:", dim(train_data), "\n")
cat("Size of validation data:", dim(validation_data), "\n")
cat("Size of test data:", dim(test_data), "\n")
```

а) **Разбиение на тренировочные данные**:

— Функция `createDataPartition` из пакета `caret` используется для генерации индексов тренировочных данных.

— Аргумент `p = 0.6` указывает, что 60% от исходных данных будет отведено на тренировочную выборку.

— Сохраняются сами данные для тренировки: `train_data`.

б) **Формирование временной выборки**:

— Оставшиеся 40% данных (не вошедшие в тренировочную выборку) сохраняются в объекте `temp_data`.

в) **Разбиение на валидационную и тестовую выборки**:

— 50% от временной выборки назначаются валидационным данным с использованием той же функции `createDataPartition`.

— Оставшиеся 50% становятся тестовыми данными.

— Данные сохраняются в `validation_data` и `test_data` соответственно.

г) **Вывод размеров выборок**:

— Используется функция `dim()` для определения количества строк и столбцов в каждой из подвыборок.

— Вывод осуществляется с помощью `cat()`.

Данные по размерности наборов предоставены ниже:

Таблица 5 — Размеры наборов данных

Набор данных	Количество наблюдений	Количество переменных
Обучающий	1226	11
Валидационный	408	11
Тестовый	406	11

Можно заметить, что сумма количества наблюдей по выборкам в точности совпадает с изначальным значением количества наблюдейний

2 Формирование softmax модели

Листинг 2.5 — Формирование модели для классификации квартир

```
1 model <- multinom(class ~ ., data = train_data)
2
3 summary(model)
```

а) **Создание модели**:

— Модель `multinom` применяется для решения многоклассовых задач классификации.

— Зависимая переменная (`class`) предсказывается на основе всех доступных признаков (`.` указывает на использование всех столбцов, кроме `class`).

— Данные для обучения берутся из `train_data`.

б) **Вывод сведений о модели**:

— Команда `summary(model)` отображает коэффициенты регрессии (β_{kj}), их стандартные ошибки и соответствующие z-значения.

— Основная информация включает:

1) **Оценки коэффициентов:** β_{kj} для каждого признака j и класса k, кроме базового.

2) **Стандартные ошибки:** величина погрешности оценки коэффициентов.

Таблица 6 — Коэффициенты и стандартные ошибки модели множественной логистической регрессии

Переменная	Коэффициенты			Стандартные ошибки		
	medium	large	huge	medium	large	huge
(Intercept)	-102.79	-91.46	-159.97	0.1659	0.5003	0.1399
price	0.0042	0.0250	0.1491	0.0054	0.0089	0.4578
nrooms	21.54	11.75	-12.42	0.6635	2.0015	0.4848
livesp	0.279	0.519	0.778	0.0449	0.0887	2.6571
kitsp	0.155	0.424	3.656	0.0794	0.1474	3.1011
dist	0.013	0.461	1.964	0.0633	0.1981	3.9800
metrdist	0.045	-0.071	2.527	0.0396	0.1364	4.0743
walk	0.107	-1.139	-12.587	0.3309	1.0230	0.3903
brick	-0.819	-2.945	-11.131	0.3915	1.1379	0.2262
floor	0.508	0.197	8.474	0.4004	2.0680	0.1399
code	0.032	0.019	0.778	0.0776	0.2517	0.9684

Также, информация о модели содержит такие показатели, как *Residual Deviance* и *AIC*. Их значения дают представление о качестве модели и её способности предсказывать целевую переменную.

Residual Deviance (Остаточная девиация)

Остаточная девиация является мерой, отражающей степень отклонения предсказаний модели от истинных значений целевой переменной.

— Расчёт остаточной девиации основан на функции логарифма правдоподобия (Log-Likelihood):

$$\text{Residual Deviance} = -2 \times \text{Log-Likelihood}.$$

— Для данной модели `Residual Deviance = 312.5433`.

— Меньшие значения остаточной девиации указывают на лучшее соответствие модели данным.

— Остаточная девиация обычно сравнивается с начальной девиацией (*Null Deviance*), которая оценивает правдоподобие модели только с учетом базового уровня (без учёта признаков).

Интерпретация

— Если модель сильно уменьшила девиацию относительно начальной (Null Deviance $\gg$ Residual Deviance), то это указывает на хороший учёт информации в данных.

— Если разница мала, модель может быть недостаточно информативной.

AIC (Критерий Акаике)

AIC (*Akaike Information Criterion*) используется для оценки качества модели с учётом её сложности.

— Формула расчёта AIC:

$$\text{AIC} = -2 \times \text{Log-Likelihood} + 2 \times k,$$

где:

Log-Likelihood — логарифм функции правдоподобия модели;

k — число параметров модели (включая смещение и коэффициенты).

— Для данной модели: `AIC = 378.5433`.

— Критерий включает штраф за увеличение числа параметров модели, чтобы избежать переобучения.

— Меньшее значение AIC указывает на лучшую модель с учётом её предсказательной силы и сложности.

Интерпретация

— AIC полезен при сравнении нескольких моделей: модель с наименьшим значением AIC предпочтительнее.

— Следует избегать исключительно снижения AIC, если это приводит к потере интерпретируемости.

2.1 Оценка модели

Листинг 2.6 — Проверка модели на валидационном наборе данных

```
validation_predictions <- predict(model, validation_data)
validation_pred_prob <- predict(model, validation_data, type="prob")
head(validation_predictions)
head(validation_pred_prob)
```

а) **Предсказание классов** (`predict`):

— Функция `predict()` используется для генерации предсказаний модели.

— Параметры:

`model` — обученная модель `multinom`.

`validation_data` — данные для валидации.

— По умолчанию, функция возвращает предсказанный класс для каждой строки валидационного набора.

— Результаты сохраняются в `validation_predictions`.

б) **Предсказание вероятностей классов** (`type="prob"`):

— Указание параметра `type="prob"` в функции `predict()` приводит к возврату вероятностей принадлежности каждого объекта к каждому классу.

— Возвращаемый результат представляет собой таблицу (матрицу), где:

Строка $i \Rightarrow$ вероятности для объекта i по всем классам.

— Результаты сохраняются в `validation_pred_prob`.

в) **Отображение результатов** (`head`):

Первый head вывел в консоль 6 раз small.

Второй:

Таблица 7 — Таблица вероятностей подсказаний класса

#	small	medium	large	huge
5	1	3.372637e-31	2.892411e-25	5.541612e-44
7	1	6.063928e-30	1.520350e-21	3.105919e-26
9	1	7.695684e-31	1.409279e-24	1.355995e-27
12	1	8.907091e-30	2.725440e-21	9.451532e-32
15	1	7.922207e-31	1.566323e-23	3.258361e-32
19	1	7.564271e-31	2.971952e-24	1.462170e-35

2.2 Оценка значимости

Листинг 2.7 — Формирование метрик значимости

```
coef_summary <- summary(model)

coefficients <- coef_summary$coefficients

estimates <- coefficients
std_errors <- coef_summary$standard.errors

alpha <- 0.05
z_alpha_over_2 <- qnorm(1 - alpha / 2)


lower_bounds <- estimates - z_alpha_over_2 * std_errors
upper_bounds <- estimates + z_alpha_over_2 * std_errors


confidence_intervals <- data.frame(
Coefficient = estimates,
Lower_Bound = lower_bounds,
Upper_Bound = upper_bounds
)


print(confidence_intervals)

z_scores <- coefficients / std_errors
p_values <- 2 * (1 - pnorm(abs(z_scores)))

```

```
cat("\nP-value:\n")
print(p_values)
```

Шаги выполнения

а) **Получение коэффициентов модели (`coef_summary$coefficients`):**

— Результаты модели извлекаются с помощью `summary(model)`. Это позволяет получить информацию о коэффициентах модели.

б) **Расчёт доверительных интервалов:**

— Доверительный интервал для каждого коэффициента рассчитывается на основе оценки коэффициента (`estimates`) и стандартной ошибки (`std_errors`).

— Используем критическое значение $z_{\alpha/2}$, которое получаем через функцию `qnorm`, основываясь на заданном уровне значимости α (например, 0.05).

— Нижний и верхний пределы доверительного интервала вычисляются как:

$$\text{Lower Bound} = \text{Coefficient} - z_{\alpha/2} \times \text{SE}$$

$$\text{Upper Bound} = \text{Coefficient} + z_{\alpha/2} \times \text{SE}$$

— Результат сохраняется в `confidence_intervals` и выводится с помощью `print`.

в) **Расчёт p-значений:**

— Для каждого коэффициента вычисляется статистика z с использованием формулы:

$$z = \frac{\text{Coefficient}}{\text{Standard Error}}$$

— После вычисления z-значений, p-значения рассчитываются как $2 \times (1 - \text{pnorm}(|z|))$, где `pnorm` вычисляет вероятность нормального распределения для заданной z-статистики.

— Вычисленные p-значения выводятся с помощью `print`.

Интерпретация

1. **Доверительные интервалы:** Для каждого коэффициента модели вычисляются доверительные интервалы, что позволяет оценить, в каком диапазоне может лежать истинное значение коэффициента с заданной вероятностью.

2. **P-значения:** P-значения указывают на статистическую значимость коэффициентов. Если p-значение меньше уровня значимости (например, $\alpha = 0.05$), то коэффициент считается статистически значимым и влияющим на модель.

Таблица 8 — Таблица коэффициентов и доверительных интервалов

	Coefficient			95% CI		
Переменная	medium	large	huge	medium	large	huge
Intercept	-102.79208	-91.46012	-159.97239	[-103.12, -102.47]	[-92.44, -90.48]	[-160.25, -159.70]
price	0.004242476	0.025008955	0.149126243	[-0.01, 0.02]	[0.01, 0.04]	[-0.75, 1.05]
nrooms	21.53842	11.75132	-12.41825	[20.24, 22.84]	[7.83, 15.67]	[-13.37, -11.47]
livesp	0.2786031	0.5188833	0.7784897	[0.19, 0.37]	[0.35, 0.69]	[-4.43, 5.99]
kitsp	0.1548881	0.4237548	3.6559646	[-0.00, 0.31]	[0.13, 0.71]	[-2.42, 9.73]
dist	0.01299013	0.46108441	1.96394778	[-0.11, 0.14]	[0.07, 0.85]	[-5.84, 9.76]
metrdist	0.04477751	-0.07134023	2.52728122	[-0.03, 0.12]	[-0.34, 0.20]	[-5.46, 10.51]
walk	0.1066931	-1.1392639	-12.5871572	[-0.54, 0.76]	[-3.14, 0.87]	[-13.35, -11.82]
brick	-0.8189447	-2.9454035	-11.1312490	[-1.59, -0.05]	[-5.18, -0.72]	[-11.57, -10.69]
floor	0.5078203	0.1969960	8.4744960	[-0.28, 1.29]	[-3.86, 4.25]	[8.20, 8.75]
code	0.03207142	0.01868934	0.77778279	[-0.12, 0.18]	[-0.47, 0.51]	[-1.12, 2.68]

Уже видно по доверительным интервалам, что модель некорректна.

Для корректировки модели определим статистически незначимые параметры с помощью P-value:

Таблица 9 — Таблица p-значений

	p-value		
Переменная	medium	large	huge
(Intercept)	0.0000000	0.0000000	0.0000000
price	0.4322993	0.0047229	0.7446396
nrooms	0.0000000	0.0000000	0.0000000
livesp	0.0000001	0.0000005	0.7695377
kitsp	0.0510531	0.0040345	0.2384270
dist	0.8375105	0.0199534	0.6216908
metrdist	0.2586784	0.6010639	0.5350591
walk	0.7471466	0.2654454	0.0000000
brick	0.0364589	0.0096421	0.0000000
floor	0.2046657	0.9241072	0.0000000
code	0.6794891	0.9408164	0.4218866

Отсюда видно, что значимыми параметрами являются только константа, **nrooms** и **livesp** (параметров для large и huge крайне мало, так что на эти метрики можно не смотреть. Убрать их нельзя, т.к. модель должна работать для классификации четрыёх классов).

Сформируем модель, основываясь только на этих параметрах.

2.3 Оптимальная модель

Листинг 2.8 — Построение оптимальной модели

```
1 model <- multinom(class ~ nrooms + livesp , data = train_data)
```

После проведения тех же процедур оценки на валидационной выборке, модель преобрела статистически значимые параметры и её точность выросла почти на процент.

Таблица 10 — Коэффициенты и стандартные ошибки модели множественной логистической регрессии

Переменная	Коэффициенты			Стандартные ошибки		
	medium	large	huge	medium	large	huge
(Intercept)	-94.29	-67.20	-62.32	0.097	0.18	0.46
nrooms	20.83	10.08	6.38	0.39	0.71	1.84
livesp	0.23	0.45	0.54	0.032	0.05	0.10

Residual Deviance: 381

AIC: 399

Хоть, разделённая разность больше, но критерий AIC показывает, что данная модель выигрывает, по сравнению с не оптимизированной (387).

Таблица 11 — P-value оптимизированной модели

Переменная	medium	large	huge
(Intercept)	0	0	0
nrooms	0.0000000000	0.0000000000	0.0005313391
livesp	3.124612e-12	0.000000e+00	1.430513e-07

Можно наблюдать, что **все** параметры являются значимимы.

Данная таблица иллюстрирует, что концы доверительных интервалов имеют один знак, и находятся на довольно малом расстоянии.

Промежуточные выводы

Удалось оптимизировать параметры модели, перейдя с 10 до 2 штук. Что позволило увеличить критерий AIC и статистически добиться значимости параметров.

Таблица 12 — Коэффициенты и доверительные интервалы модели множественной логистической регрессии

Переменная	Коэффициенты			Доверительные интервалы	
	medium	large	huge	Lower Bound	Upper Bound
Intercept	-94.28639	-67.20221	-62.32092	-94.47669	-94.09608
nrooms	20.832198	10.076997	6.376026	20.070977	21.593419
livesp	0.2272280	0.4462840	0.5451541	0.1633498	0.7482380

Следующим шагом будет являться использование модели на тестовых данных.

2.4 Финальные тест

Для демонстрации результатов, модели будет передан тестовый набор параметров. На основании анализа результатов можно будет сделать выводы о точности предсказаний.

Матрица путаницы и статистики классификации

Листинг 2.9 — Формирование матрицы путаницы

```
1 misclassification<-caret::confusionMatrix(data=test_predictions
    ,reference=test_data$class)
2 misclassification
```

Матрица путаницы

Результаты классификации модели:

Prediction \ Reference	**small**	**medium**	**large**	**huge**
small	326	12	0	0
medium	8	52	5	0
large	0	1	1	0
huge	0	0	1	0

Общая статистика

— **Accuracy (Точность):** 93.35%
— **95% Доверительный интервал (CI):** $(90.47\%, 95.57\%)$
— **No Information Rate (NIR):** 82.27%
— **P-Value [Acc > NIR]:** 5.826×10^{-11}
— **Коэффициент Kappa:** 0.7702
— **Тест МакНемара:** Недоступен (NA)

Статистика по классам

Таблица ниже содержит метрики для каждого класса:

Метрика	**small**	**medium**	**large**	**huge**
Sensitivity	0.9760	0.8000	0.1429	NA
Specificity	0.8333	0.9619	0.9975	0.9975
Pos Pred Value	0.9645	0.8000	0.5000	NA
Neg Pred Value	0.8824	0.9619	0.9851	NA
Prevalence	0.8227	0.1601	0.0172	0.0000
Detection Rate	0.8030	0.1281	0.0025	0.0000
Detection Prevalence	0.8325	0.1601	0.0049	0.0025
Balanced Accuracy	0.9047	0.8809	0.5702	NA

Интерпретация

— **Чувствительность (Sensitivity):** Показывает, какая доля объектов каждого класса была правильно классифицирована.

— **Специфичность (Specificity):** Указывает на способность модели корректно классифицировать объекты, которые не принадлежат к рассматриваемому классу.

— **Сбалансированная точность (Balanced Accuracy):** Учитывает баланс между чувствительностью и специфичностью для каждого класса.

— **Прогностическая ценность (PPV, NPV):** PPV показывает вероятность того, что предсказанный класс является правильным, а NPV — что объект не принадлежит к рассматриваемому классу.

Результаты анализа ROC-кривых и значений AUC

Листинг 2.10 — Построение и анализ ROC-кривых

```
test_pred_prob <- predict(model, test_data, type = "prob")

roc_list <- list()
auc_list <- c()

for (class in colnames(test_pred_prob)) {
    binary_true_labels <- ifelse(test_data$class == class, 1, 0)


    if (length(unique(binary_true_labels)) == 2) {
        roc_curve <- roc(binary_true_labels, test_pred_prob[, class],
            quiet = TRUE)
        roc_list[[class]] <- roc_curve

        auc_list <- c(auc_list, auc(roc_curve))

        cat(paste("Class:", class, "- AUC:", round(auc(roc_curve), 3)),
            "\n")
    } else {
        cat(paste("Class:", class, "- not enought data for draw
            ROC-curve\n"))
    }
}
plot(roc_list[[1]], col = "blue", main = "ROC-curve (One-vs-Rest)", lwd =
    2)
for (i in 2:length(roc_list)) {
    plot(roc_list[[i]], col = i, add = TRUE, lwd = 2)
}
legend("bottomright", legend = colnames(test_pred_prob), col =
    1:length(roc_list), lwd = 2)
```

Ниже представлены результаты вычислений ROC-кривых и площадей под кривой (AUC) для каждого класса целевой переменной:

Класс	AUC
small	0.989

medium	0.981
large	0.996
huge	**Недостаточно уровней для построения ROC-кривой**

Описание результатов

— **Класс small:** Значение AUC составило 0.989, что свидетельствует о высокой способности модели различать данный класс от остальных.

— **Класс medium:** Значение AUC составляет 0.981, что также указывает на высокую точность классификации данного класса.

— **Класс large:** Наивысшее значение AUC среди классов — 0.996, Но этот показатель вызван нехваткой данных, т.к. выборка крайне неоднородна.

— **Класс huge:** Для данного класса не удалось построить ROC-кривую, так как в целевых данных отсутствовали оба уровня (0 и 1), необходимых для анализа.

Также, приведём визуализацию предсказанных классов в срезе **class** и **nrooms**

2.5 Вывод по классификации softmax

По результатам построения модели можно сказать, что она с очень хорошей точностью предсказывает классификацию для квартир класса small и medium. Но для класса **large** модель работает хуже, а предсказание класса **huge** очень плохое.

Такое поведение объясняется неоднородностью данных в датасете.

Таким образом, оптимальным решением будет использовать модель для бинарной классификации в рамках множества {small, medium}.

3 SVM-модель

В качестве SVM моедли будем использовать ksvm-модель из пакета **kernlab**.

Список использованных источников

1. *Кривоносова, А.Ф.* Численные методы в алгебре / А.Ф. Кривоносова. — Москва: Наука, 2005.

2. *Исаев, А.А.* Численные методы линейной алгебры / А.А. Исаев, Г.П. Копытов. — Москва: Физматлит, 2011.

3. *Колобов А. Г., Молчанова Л.А.* Численные методы линейной алгебры / Молчанова Л.А. Колобов А. Г. — 1th edition. — Владивосток: Издательство Дальневосточного университета, 2008.