Национальный исследовательский университет «Высшая школа экономики»
Факультет компьютерных наук
Образовательная программа: Прикладная математика и информатика

Отчет по Домашнему заданию №1
по майнору «Прикладной статистический анализ»

«Предсказание стоимости автомобиля с помощью регрессионного анализа»

Работу выполнила
студентка 3 курса
Ульянова Александра Игоревна

Москва, 2023 г.

Содержание

Введение

Для анализа мной был выбран датасет под названием “Car Price Prediction”. В нем приведена информация о различных параметрах машин и их стоимости. Всего в датасете представлена информация о 205 машинах, каждая их которых характеризуется 26 признаками. Набор данных включает в себя такую информацию, как название автомобиля, его размеры, характеристики двигателя, число дверей и многое другое.

Среди признаков присутствует признак под названием “price”. Он означает стоимость данного автомобиля и является непрерывным. Задачей данного исследования будет научиться прогнозировать его при помощи остальных переменных. Это задача может быть применима на практике. Так, например, владельцу автомобиля было бы полезно знать, за какую стоимость он может ее продать исходя из её параметров. Аналогично и для покупателя: имея модель перед глазами он может оценить, сколько денег ему потребуется для покупки автомобиля с желаемыми техническими характеристиками. Может так же обнаружиться, что какой-то признак сильно влияет на зависимую переменную. Например, автомобили, заправляемые дизельным топливом, могут стоить дороже, чем автомобили, ездящие на обычном бензине. Или же возможно расположение двигателя в автомобиле сильно влияет на его стоимость.

Данные были взяты с сайта kaggle.com.

Цель исследования: научиться прогнозировать стоимость автомобиля по его параметрам

Задачи исследования:

- Поиск и сбор необходимых данных
- Первичный анализ данных
- Корреляционный анализ данных
- Построение регрессионной модели
- Проверка гомоскедастичности ошибок
- Построение Ridge-регрессии
- Построение Lasso-регрессии
- Сравнение полученных моделей и выбор наилучшей
- Подведение итогов

Первичный анализ данных

Выбор объясняющих переменных и подготовка данных

Посмотрим на список и тип всех переменных, которые имеются в нашем датасете.

car_ID: Id автомобиля, числовая переменная
symboling: обозначение, числовая переменная
CarName: марка и модель автомобиля, категориальная переменная
Fueltype: тип топлива, категориальная
aspiration: турбонаддув в двигателе
doornumber: число дверей, изначально категориальная
carbody: вид тела авто, категориальная
drivewheel: тип ведущего колеса, категориальная
enginelocation: место расположения двигателя, категориальная
wheelbase: колесная база, числовая
carlength: длина автомобиля, числовая
carwidth: ширина автомобиля, числовая
carheight: высота автомобиля, числовая
curbweight: собственная масса автомобиля, числовая
enginetype: вид двигателя, категориальная
cylindernumber: число цилиндров, изначально категориальная
enginesize: объем двигателя, числовая
fuelsystem: топливная система, категориальная
boreratio: что-то про диаметр поршня, числовая
stroke: число тактов двигателя, числовая
compressionratio: степень сжатия, числовая
horsepower: число лошадиных сил автомобиля, числовая
peakrpm: максимальное число оборотов в минуту, числовая
citympg: число миль на галлон в городе, числовая
highwaympg: число миль на галлон на трассе, числовая
price: стоимость, числовая

Заметим, что у нас есть несколько категориальных признаков. Некоторые из них (fueltype, aspiration и enginelacation) принимают только два различных значения. Это значит, что мы можем преобразовать эти переменные в бинарные, такие как fueltype_diesel, aspiration_std и enginelocation_front. Такое преобразование позволит нам избавиться от их категориального типа, и, следовательно, спокойно работать с этими признаками при построении линейной модели. Еще две категориальные переменные (doornumber и cylindernumber) легко преобразовываются в числовые. Сейчас их значения записаны в текстовом виде, например ‘two’ или ‘four’, так что просто преобразуем их в соответствующие числа для дальнейшей работы. Оставшиеся категориальные признаки можно было бы преобразовать в бинарные с помощью one-hot encoding. Однако в таком случае число переменных в нашем датасете бы сильно “раздулось” и количество бинарных переменных превысило бы 20%. Мы хотим этого избежать по условию задания, так что просто избавимся от этих переменных.

Также нам потребуется удалить признаки car_ID и symbolling, поскольку они чисто технические и к нашим автомобилям прямого отношения не имеют. Они будут только портить нашу линейную модель.

Все остальные признаки числовые и непрерывные, мы будем рассматривать их как независимые.

Предположения

Сделаем предположения о том, как именно независимые переменные могут влиять на объясняемую переменную. Во-первых, параметры двигателя вероятно будут сильно влиять на его стоимость. Чем мощнее двигатель, чем больше в нем лошадиных сил и чем больше объем, тем вероятно дороже автомобиль. Во-вторых, габариты автомобиля скорее всего влияют на его цену. Предполагаю, что чем больше длина, ширина и высота автомобиля, тем более он дорогой (как например джип бывает дороже легкового авто). Все эти предположения мы сможем проверить в ходе дальнейшего анализа.

Дескриптивный анализ данных

Теперь более подробно посмотрим на наши данные. После предварительной обработки данных (удаления ненужных признаков, преобразований, удаления дубликатов) в нашей таблице содержится информация о 203 автомобилях, каждый из которых описан 19-ю признаками.

С помощью команды $df.describe()$ получим основную информацию о каждом из признаков. Результаты представлены ниже на Рис.1

	doornumber	wheelbase	carlength	carwidth	carheight	curbweight	cylindernumber	enginesize	boreratio	stroke	compressionratio
count	203.000000	203.000000	203.000000	203.000000	203.000000	203.000000	203.000000	203.000000	203.000000	203.000000	203.000000
mean	3.123153	98.806404	174.214286	65.928571	53.754680	2560.571429	4.384236	127.231527	3.333005	3.254877	10.158719
std	0.994841	6.030353	12.284573	2.145470	2.436869	520.717792	1.085525	41.716874	0.270160	0.314998	3.987270
min	2.000000	86.600000	141.100000	60.300000	47.800000	1488.000000	2.000000	61.000000	2.540000	2.070000	7.000000
25%	2.000000	94.500000	166.800000	64.150000	52.000000	2179.500000	4.000000	97.500000	3.150000	3.110000	8.600000
50%	4.000000	97.000000	173.200000	65.500000	54.100000	2420.000000	4.000000	120.000000	3.310000	3.290000	9.000000
75%	4.000000	102.400000	183.300000	66.900000	55.500000	2943.500000	4.000000	143.000000	3.585000	3.410000	9.400000
max	4.000000	120.900000	208.100000	72.300000	59.800000	4066.000000	12.000000	326.000000	3.940000	4.170000	23.000000

	horsepower	peakrpm	citympg	highwaympg	price	fueltype_diesel	aspiration_std	enginelocation_front
count	203.000000	203.000000	203.000000	203.000000	203.000000	203.000000	203.00000	203.000000
mean	104.305419	5121.428571	25.197044	30.719212	13337.633828	0.098522	0.82266	0.985222
std	39.657386	477.873206	6.561257	6.901354	8003.995399	0.298756	0.38290	0.120963
min	48.000000	4150.000000	13.000000	16.000000	5118.000000	0.000000	0.00000	0.000000
25%	70.000000	4800.000000	19.000000	25.000000	7793.500000	0.000000	1.00000	1.000000
50%	95.000000	5200.000000	24.000000	30.000000	10345.000000	0.000000	1.00000	1.000000
75%	116.000000	5500.000000	30.000000	34.000000	16509.000000	0.000000	1.00000	1.000000
max	288.000000	6600.000000	49.000000	54.000000	45400.000000	1.000000	1.00000	1.000000

Рисунок 1. Основная информация о признаках

Далее воспользуемся функцией $df.skew(numeric\ only = True)$, чтобы получить информацию о скошенности данных. Это мера асимметрии

распределения данных, которая позволяет оценить, насколько данные в каждом столбце отклоняются от нормального распределения. Обычно, если скошенность больше 1 или меньше -1, это считается существенной скошенностью данных. Посмотрим на нашу целевую переменную. Её скошенность составляет 1.76, что существенно больше 1. Значит распределение нашей целевой переменной смещено вправо (в сторону более высоких значений) и имеет длинный правый хвост. Со скошенностью остальных переменных можно ознакомиться ниже на Рис. 2

```
df.skew(numeric_only=True)
```

```
doornumber            -0.250046
wheelbase              1.037060
carlength              0.144192
carwidth               0.892857
carheight              0.046410
curbweight             0.668924
cylindernumber         2.799022
enginesize             1.938233
boreratio              0.002711
stroke                -0.682563
compressionratio       2.595900
horsepower             1.396628
peakrpm                0.094078
citympg                0.671487
highwaympg             0.550134
price                  1.767101
fueltype_diesel        2.714405
aspiration_std        -1.702117
enginelocation_front  -8.102485
dtype: float64
```

Рисунок 2. Скошенность численных переменных

Теперь посмотрим информацию о эксцессе данных в данном DataFrame с помощью функции $df.kurtosis()$. Эксцесс - это мера, которая оценивает, насколько данные имеют более крутые или плоские хвосты (тяжелые или легкие хвосты) по сравнению с нормальным распределением. Значение равное 3 указывает на нормальное распределение. Наша зависимая переменная имеет значение эксцесса равное 3.007, что означает, что она имеет распределение очень очень близкое к нормальному. Со значениями для остальных переменных можно ознакомиться на Рис.3

```
df.kurtosis(numeric_only=True)
```

```
doornumber            -1.956855
wheelbase              0.994287
carlength             -0.045724
carwidth               0.693304
carheight             -0.416549
curbweight            -0.051021
cylindernumber        13.556088
enginesize             5.264396
boreratio             -0.764052
stroke                 2.129056
compressionratio       5.146039
horsepower             2.645449
peakrpm                0.093533
citympg                0.578904
highwaympg             0.448895
price                  3.007583
fueltype_diesel        5.421314
aspiration_std         0.906032
enginelocation_front  64.283535
dtype: float64
```

Рисунок 3. Коэффициенты эксцесса

Также построим boxplot, чтобы лучше понять распределение нашей целевой переменной. Из Рис.4 видно, что у нас имеется достаточно много выбросов с большими значениями. Это соответствует полученной нами ранее информации о тяжелом правом хвосте распределения целевой переменной.

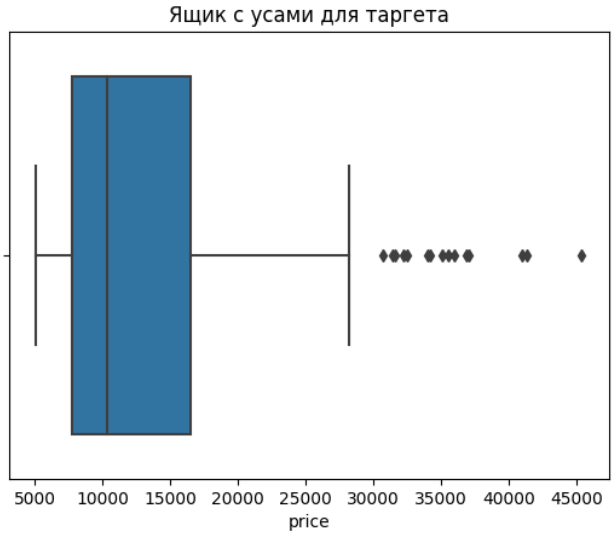

Рис.4 Ящик с усами для целевой переменной price

Теперь построим QQ-plot. Quantile-Quantile plot - это визуальный способ оценки того, насколько данные соответствуют нормальному распределению. Если точки на QQ-графике приближаются к прямой линии (45-градусной линии), это указывает на то, что данные близки к нормальному распределению. Из Рис.5 ниже следует, что распределение целевой переменной отдаленно похоже на нормальное, но все же не до конца, поскольку до 45-градусной линии не доходит.

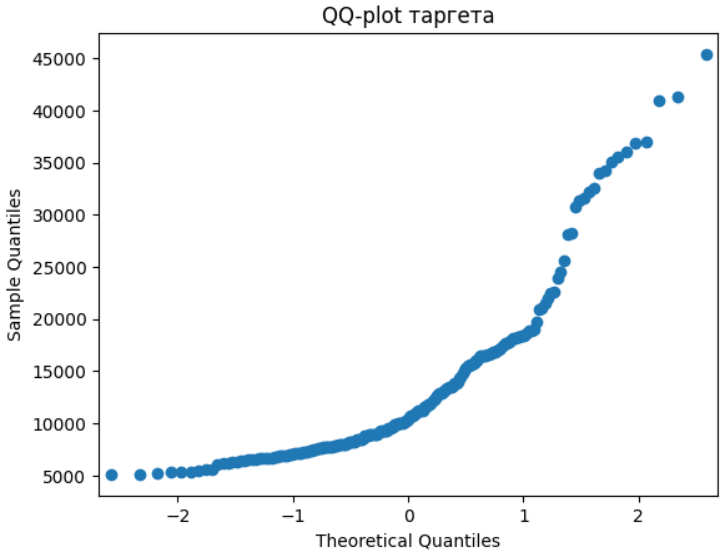

Рис.5 QQ-plot для price

Для лучшего понимания остальных переменных построим еще несколько графиков. Во-первых, посмотрим на распределение числа цилиндров в двигателе, для чего построим гистограмму. Результат представлен ниже на Рис. 6. Можно заметить, что в основном представлены автомобили с 4-ёх цилиндровыми

двигателями. Присутствует также достаточное количество автомобилей с 6 цилиндрами. Автомобилей с другим числом цилиндров очень мало.

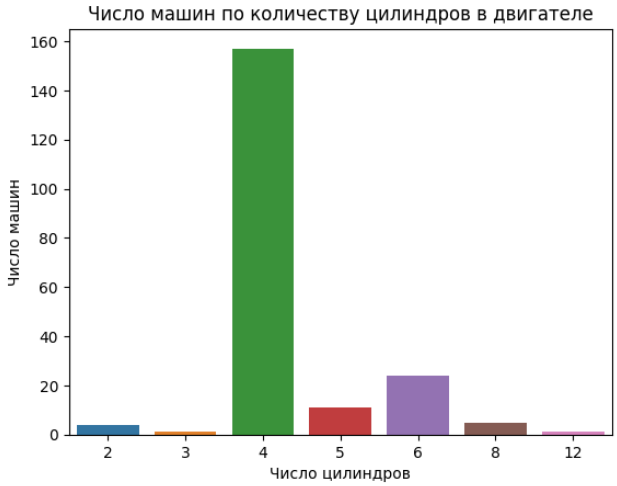

Рисунок 6. Распределение машин по числу цилиндров

Посмотрим еще на распределение машин по ширине и длине. Для этого тоже изобразим гистограммы, они приведены на Рис. 7. Видно, что длина машин распределена почти нормально и в среднем составляет около 173 см. Гистограмма с шириной машин гораздо меньше напоминает нормальное распределение. Большинство значений здесь начинается с 63 см и образует хвост справа до 72 см.

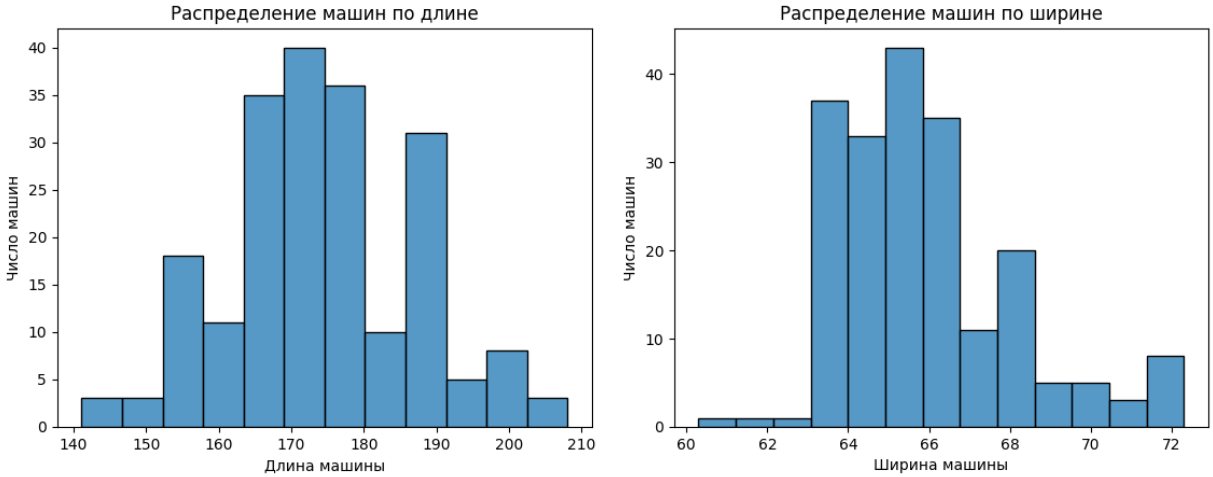

Рисунок 7. Распределение машин по длине и ширине

Еще посмотрим на один из бинарных параметров машины – fueltype_diesel. Он показывает, завравляют ли данную машину дизелем или газом. Его распределение представлено на Рис. 8. По данному графику можно наблюдать, что в основная часть машин в датасете использует в качестве топлива газ.

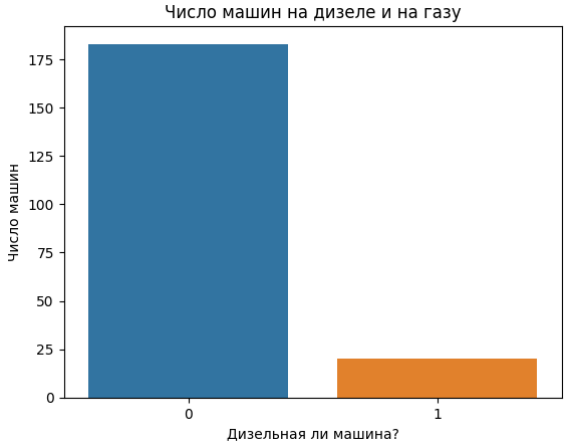

Рисунок 8. Распределение параметра fueltype_diesel

Не будем строить графики для остальных переменных ввиду большого их количества. Основное представление о наших данных мы уже получили.

Предположения

Проведя первичный анализ данных, попробуем предположить, каким образом независимые переменные должны оказывать влияние на объясняемую переменную.

Во-первых, размеры машины (а именно её длина, ширина и высота) будут прямо пропорциональны стоимости автомобиля. Это становится понятно, если вспомнить, что легковые автомобили обычно стоят дешевле спортивных джипов или грузовых автомобилей. Во-вторых, параметры двигателя будут иметь прямую положительную связь с ценой автомобиля. Чем мощнее двигатель, тем более он дорогой, а соответственно это прибавляет стоимости и автомобилю. Аналогично и с типом топлева. Автомобили ездящие на газу наверняка стоят дешевле автомобилей, использующих дизельное топливо.

Корреляционный анализ данных

Теперь проведем корреляционный анализ данных. Он покажет нам как связаны переменные между собой. Начнем с построения корреляционной матрицы. Эта матрица содержит коэффициенты корреляции между парами переменных и позволяет исследовать, насколько эти переменные взаимосвязаны. Часть полученной матрицы представлена на Рис.9, с полной версией можно ознакомиться в файле с расчетами.

	doornumber	wheelbase	carlength	carwidth	carheight	curbweight	cylindernumber	enginesize	boreratio	stroke
doornumber	1.000000	0.450172	0.402621	0.208016	0.558155	0.199687	-0.016530	0.020900	0.120368	-0.008403
wheelbase	0.450172	1.000000	0.874344	0.793538	0.585640	0.774721	0.337969	0.566513	0.483901	0.163052
carlength	0.402621	0.874344	1.000000	0.839592	0.482457	0.876857	0.430145	0.681101	0.600004	0.133155
carwidth	0.208016	0.793538	0.839592	1.000000	0.270516	0.865861	0.544524	0.733554	0.554055	0.185611
carheight	0.558155	0.585640	0.482457	0.270516	1.000000	0.287119	-0.018462	0.058049	0.158481	-0.053710
curbweight	0.199687	0.774721	0.876857	0.865861	0.287119	1.000000	0.609650	0.849584	0.644433	0.170993
cylindernumber	-0.016530	0.337969	0.430145	0.544524	-0.018462	0.609650	1.000000	0.846453	0.228996	0.008826
enginesize	0.020900	0.566513	0.681101	0.733554	0.058049	0.849584	0.846453	1.000000	0.580263	0.204990
boreratio	0.120368	0.483901	0.600004	0.554055	0.158481	0.644433	0.228996	0.580263	1.000000	-0.054518
stroke	-0.008403	0.163052	0.133155	0.185611	-0.053710	0.170993	0.008826	0.204990	-0.054518	1.000000

Рисунок 9. Часть корреляционной матрицы

Коэффициент корреляции — это показатель степени связи между двумя переменными или измерениями. Коэффициент корреляции изменяется от -1 до +1. Величина коэффициента корреляции по модулю показывает степень зависимости. r=0 — нет никакой связи; r =0.01–0.3 — слабая связь; r =0.31–0.7 — умеренная связь; r =0.71–0.99 — сильная связь; r =1 — совершенная связь. Положительное или отрицательное значение коэффициента говорит о наличии прямой или обратной связи между переменными. В нашем случаем мы видим сильную положительную связь между enginesize и curbweight (0.849), enginesize и cylindernumber (0.846), enginesize и carwidth(0.733554), enginesize и horsepower (0.809792), а также некоторыми другими параметрами двигателя. Это вполне объяснимо: если двигатель большой, то он обычно мощный, с большим количеством цилиндров и подходит для тяжелых автомобилей. Также сильная положительная связь наблюдается между размерностями автомобиля (длиной, шириной и высотой). Это логично, ведь производители стараются соблюдать стандартные пропорции автомобиля.

Между значениями остальных пар переменных есть только слабые взаимосвязи.

Далее построим попарные диаграммы рассеяния для каждой пары признаков. Диаграмм получится очень много, так что все их включать в отчет я не буду, однако рассмотрю наиболее интересные. Взглянем, например, на диаграмму, приведенную на Рис. 10. Она показывает зависимость цены автомобиля от размера его двигателя. Видно, что между переменными присутствует сильная прямая зависимость: чем больше становится размер двигателя, тем более дорогой автомобиль.

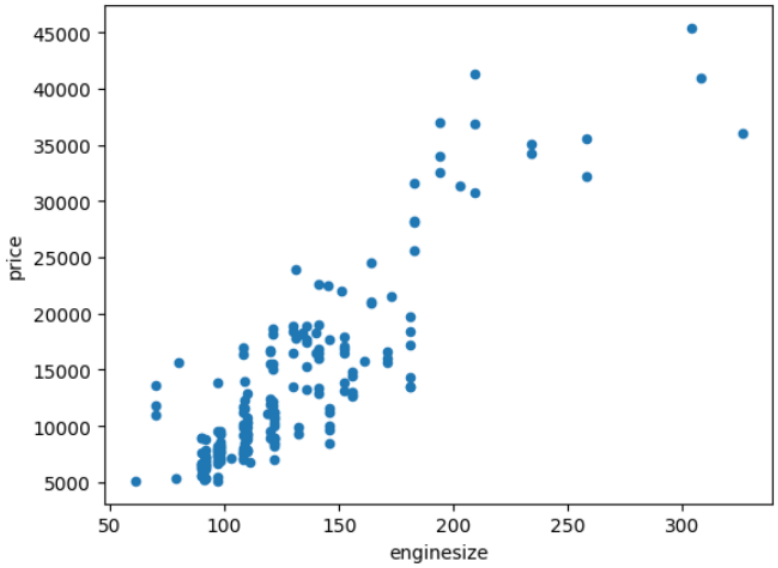

Рисунок 10. Зависимость цены автомобиля от размера двигателя

Посмотрим также на зависимость стоимости авто от числа цилиндров на Рис. 11. График говорит нам о том, что чем больше цилиндров – тем дороже автомобиль. Так, например, 4-ех цилиндровые автомобили в среднем дешевле 6-ти цилиндровых.

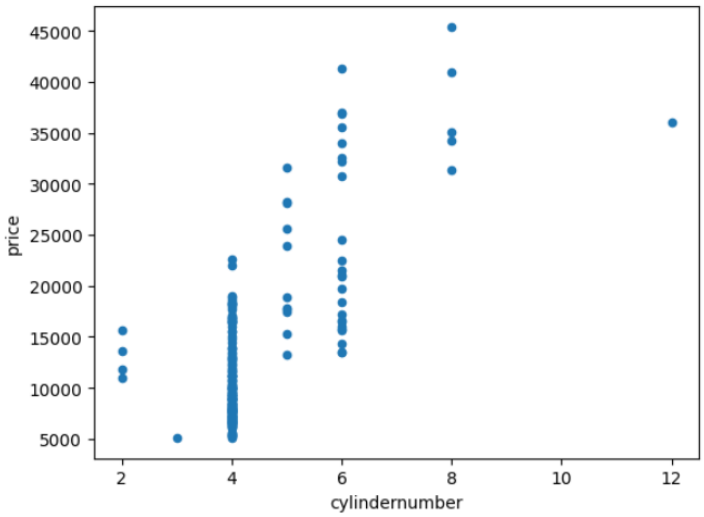

Рисунок 11. Зависимость стоимость авто от числа цилиндров

Мультиколлинеарность

Variance Inflation Factor (VIF), или фактор инфляции дисперсии, является статистическим показателем, который используется для оценки мультиколлинеарности в множественной регрессии. Мультиколлинеарность возникает, когда две или более независимые переменные в модели имеют сильную линейную зависимость между собой. VIF вычисляется для каждой независимой переменной в множественной регрессии и показывает, во сколько раз дисперсия коэффициента регрессии данной переменной больше, чем она была бы, если переменные были бы независимыми. Полученные значения VIF представлены ниже на Рис. 12

	признак	VIF		признак	VIF
0	doornumber	18.968694	9	stroke	200.719439
1	wheelbase	2047.603215	10	compressionratio	611.786266
2	carlength	2128.401384	11	horsepower	98.450379
3	carwidth	3482.396557	12	peakrpm	273.060352
4	carheight	1045.629644	13	citympg	490.455320
5	curbweight	444.077502	14	highwaympg	573.757617
6	cylindernumber	198.839260	15	fueltype_diesel	99.938812
7	enginesize	267.080015	16	aspiration_std	17.600923
8	boreratio	550.504365	17	enginelocation_front	112.446498

Рисунок 12. Значения VIF

Высокие значения VIF указывают на сильную мультиколлинеарность. В нашем случае высокая мультиколлинеарность присутствует у 4 признаков: wheelbase, carlength, carwidth, carheight. Удалим 3 из этих четырех признаков. Значения VIF после удаления можно увидеть на Рис. 13. Мы видим, что у остальных признаков значения VIF уменьшились, что очень хорошо. Больше ничего удалять не будем, ведь значения не слишком большие не будут сильно портить нашу будущую линейную модель.

	признак	VIF
0	doornumber	16.295024
1	wheelbase	966.503215
2	curbweight	391.683457
3	cylindernumber	150.042528
4	enginesize	222.349694
5	boreratio	371.374206
6	stroke	170.958958
7	compressionratio	571.940000
8	horsepower	95.937071
9	peakrpm	244.036476
10	citympg	448.709588
11	highwaympg	531.402594
12	fueltype_diesel	93.975090
13	aspiration_std	17.263917
14	enginelocation_front	104.580928

Рисунок 13. Значения VIF после удаления 3 признаков

Линейная регрессия

Построим уравнение линейной регрессии для предсказания price по переменным предикторам. Для этого воспользуемся стандартной моделью LinearRegression из библиотеки sklearn. Для начала разобъем наши данные на тренировочную и тестовую выборку в соотношении 7:3. После этого выделим отдельно x_train, x_test, y_train и y_test. Будем также использовать StandardScaler для масштабирования числовых признаков. Это поможет улучшить качество нашей модели, а также сделает коэффициенты при каждом из признаков более интерпретируемыми. Теперь возьмём модель, обучим её и посмотрим ошибку на тестовых данных.

```
scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)

model = LinearRegression()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
print("Test RMSE = %.4f" % mean_squared_error(y_test, y_pred, squared=False))
print("Test MSE = %.4f" % mean_squared_error(y_test, y_pred))
print('R2-score', r2_score(y_test, y_pred))
print('MAE', mean_absolute_error(y_test, y_pred))
```

```
Test RMSE = 3402.2746
Test MSE = 11575472.2344
R2-score 0.8475811243262594
MAE 2563.090771813641
```

Рисунок 14. Построение линейной модели

Мы видим, что значение MAE, RMSE и MSE получилось очень большим. Однако это связанно не с тем, что наша модель плоха, а лишь с большими значениями нашего таргета. В данном случае более показательно значение R2, будем смотреть на него. В данном случае оно равно 0.84, что достаточно близко к единице. Этого говорит о том, что качество нашей модели вполне хорошее. Но быть может логарифмирование таргета сделает нашу модель еще лучше.

Логарифмирование таргета

Попробуем прологарифмировать наш таргет и построить модель, которая будет предсказывать именно этот логарифмированный признак. Возможно это положительно скажется на нашей модели, уменьшит ошибки.

```
scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)

model = LinearRegression()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
print("Test RMSE = %.4f" % mean_squared_error(y_test, y_pred, squared=False))
print("Test MSE = %.4f" % mean_squared_error(y_test, y_pred))
print('R2-score', r2_score(y_test, y_pred))
print('MAE', mean_absolute_error(y_test, y_pred))
```

```
Test RMSE = 0.2027
Test MSE = 0.0411
R2-score 0.8564278950740011
MAE 0.1527428596591534
```

Рисунок 14. Построение линейной модели с логарифмированным таргетом

Как мы видим значения ошибок стали гораздо более адекватными и интерпретируемыми. И значение R2-score теперь составляет 0.85 вместо 0.84 на нелогарифмированном таргете. Оно ближе к единице, а значит модель с логарифмированным таргетом чуть лучше и проводить дальнейшие вычисления лучше на ней.

В итоге у нас получился такой вектор весов модели:

```
[ 0.00079511, 0.07904059, 0.1166798 , -0.11135108, 0.20482105, -
0.04564986, -0.0494207 , 0.12959726, 0.13187487, 0.03250628, -0.23561692,
0.13217387, -0.02752886, 0.00075548, -0.05541958]
```

а значит уравнение линейной регрессии имеет вид

$$logprice = 0.00079511 * doornumber + 0.07904059 * wheelbase + 0.1166798 * curbweight - 0.11135108 * cylindernumber + \ldots$$

Видно, что в нашем уравнении присутствуют околонулевые коэффициенты, которые почти не вносят вклад в ответ, но сильно усложняют модель. Избавиться от них нам поможет позже Lasso-регрессия.

Гетероскедастичность и гомоскедастичность ошибок

Чтобы убедиться в адекватности нашей модели (с логарифмированным таргетом) проверим гетероскедастичность ошибок. Гетероскедастичность означает, что дисперсия ошибок модели не остается постоянной по всем значениям предикторов, что может привести к недопустимым результатам при оценке параметров модели. Это может привести также к неверным выводам относительно статистической значимости коэффициентов.

Для данной проверки построим график зависимости остатков дисперсии от предсказанных значений. Результат построения приведен ниже на Рис. 15. По нему не наблюдается никакого тренда в изменении дисперсии остатков, а значит хочется сделать вывод, что гетероскедастичности ошибок в данном случае нет.

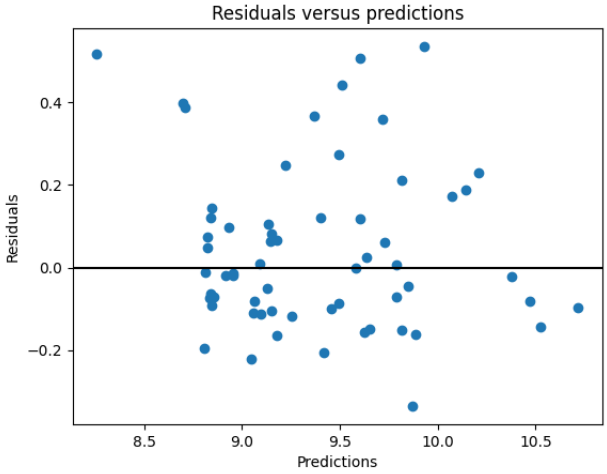

Рис. 15 Зависимость остатков дисперсии от предсказаний

Напротив, гомоскедастичность ошибок в контексте статистического анализа означает, что дисперсия ошибок модели остаётся постоянной по всем значениям предикторов. Для проверки гомоскедастичности ошибок проведем тесты Бройша-Пагана и Уайта.

Нулевой гипотезой для теста Бройша-Пагана будет являться гомоскедантичность ошибок. Выберем также уровень значимости 0.05 Для проведения теста Бройша-Пагана используем функцию `het_breuschpagan` из библиотеки `statsmodels.stats.diagnostic`. Результаты теста приведены на Рис. 16. Видно, что значение p-value меньше 0.05, а значит на уровне значимости 0.05 мы отвергаем нулевую гипотезу. Вывод: наши ошибки гетероскедантичны.

```
from statsmodels.stats.diagnostic import het_breuschpagan

x_train_new = sm.add_constant(x_train)
model = sm.OLS(y_train, x_train_new).fit()
bp_test = het_breuschpagan(model.resid, model.model.exog)

labels = ['Статистика теста Лагранжа', 'p-value', 'F-Statistic', 'F-Test p-value']

for name, value in zip(labels, bp_test):
    print(f"{name}: {value}")
```

```
Статистика теста Лагранжа: 26.55529834520424
p-value: 0.03257557534001872
F-Statistic: 1.9322195207080695
F-Test p-value: 0.025681053196682724
```

Рис. 16 Тест Бройша-Пагана

Аналогичным образом проведем и тест Уайта. Его результаты можно увидеть на Рис.17. Значение тестовой статистики равно 130, что достаточно велико и может свидетельствовать о гетероскедастичности ошибок. Значение p-value составляет 0.019. Это вероятность получить такие или более экстремальные результаты при условии, что гипотеза о гомоскедастичности верна. Это значение меньше уровня значимости 0.05, значит тоже есть гетероскедастичность.

```python
from statsmodels.stats.diagnostic import het_white

x_train_new = sm.add_constant(x_train)
model = sm.OLS(y_train, x_train_new).fit()
white_test = het_white(model.resid , model.model.exog)

labels = ['Test Statistic', 'Test Statistic p-value', 'F-Statistic', 'F-Test p-value']

for name, value in zip(labels, white_test):
    print(f"{name}: {value}")
```

```
Test Statistic: 130.05498661307288
Test Statistic p-value: 0.2111219926907011
F-Statistic: 2.1221994450217596
F-Test p-value: 0.019733324466537635
```

Рис. 17. Тест Уайта

Оба теста указывают на некоторые признаки гетероскедастичности, однако уровень значимости различен. Тест Бройша-Пагана показывает более низкое p-значение, что может говорить в пользу более сильной уверенности в наличии гетероскедастичности.

Так как наши ошибки оказались гетероскедастичны, то применим поправку Уайта. Результаты применения представлены на Рис. 18. Теперь R2-score нашей модели составляет 0.896, что сильно лучше предыдущих полученных нами результатов.

OLS Regression Results

Dep. Variable:	log_price	R-squared:	0.896
Model:	OLS	Adj. R-squared:	0.884
Method:	Least Squares	F-statistic:	197.5
Date:	Mon, 27 Nov 2023	Prob (F-statistic):	8.79e-80
Time:	09:18:05	Log-Likelihood:	62.251
No. Observations:	142	AIC:	-92.50
Df Residuals:	126	BIC:	-45.21
Df Model:	15		
Covariance Type:	HC1		

Рис. 18. Поправка Уайта

Анализ ошибок

Для анализа ошибок модели будем визуализировать остатки регрессии и делать выводы из графиков. Для начала построим обычную гистограмму (см. рисунок 18). Она в целом напоминает нормальное распределение со средним равным нулю.

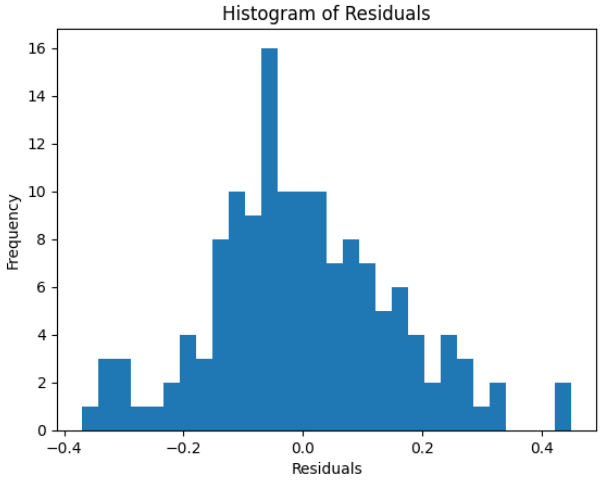

Рис. 18 Распределение остатков регрессии

Схожесть с нормальным распределением можно также пронаблюдать, построив QQ-plot остатков регрессии. Синие точки практически повторяют собой красную линию.

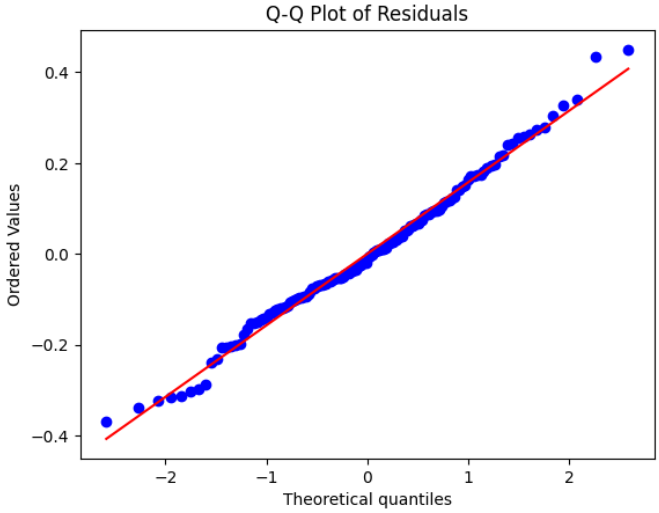

Рис. 19 QQ-plot для остатков регрессии

Так как остатки распределены нормально, то параметры модели, оцененные методом наименьших квадратов (OLS), будут несмещенными и эффективными. Нормальное распределение остатков также обеспечивает адекватность для точечных прогнозов, так как предполагается, что ошибки прогноза также будут распределены нормально.

Ridge-регрессия и кросс-валидация

Тепень построим модель получше, используя Ridge-регрессию. Ridge-регрессия добавляет L2-регуляризацию к линейной регрессии, добавляя к функции потерь сумму квадратов коэффициентов модели умноженных на параметр α. Это штрафует модель за большие значения коэффициентов, что помогает предотвратить переобучение. Большое значение α увеличивает силу регуляризации. Формула для функции потерь следующая:

$$Q_\alpha(w) = \|y - Xw\|^2 + \alpha\|w\|_2$$

Для улучшения точности модели также будем использовать кросс-валидацию с 3 фолдами. Соответствующий код представлен ниже на Рис. 20.

```
kf = KFold(n_splits=3)
for train_index, test_index in kf.split(X):
    x_train, x_test = x.iloc[train_index], x.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]

    scaler = StandardScaler()
    x_train = scaler.fit_transform(x_train)
    x_test = scaler.transform(x_test)

    ridge_model = Ridge()
    ridge_model.fit(x_train, y_train)
    y_pred = ridge_model.predict(x_test)
    mae_errors.append(mean_absolute_error(y_pred, y_test))
    r2_errors.append(r2_score(y_test, y_pred))


print('Test mean MAE:', np.mean(mae_errors))
print('Test mean R2-score:', np.mean(r2_errors))
```

```
Test mean MAE: 0.16681407424127978
Test mean R2-score: 0.7982770070463339
```

Рис. 20 Ridge-регрессия с кросс-валидацией

Из построенных с помощью кросс-валидации 3-ех моделей мы получили средний R2-score равный 0.79. Этот показатель меньше, а значит хуже, чем было у нас ранее в линейной модели.

Lasso-регрессия и кросс-валидация

Попробуем избавиться от весов близким к нулю в нашей модели, тем самым сделаем ее легче. Для этого используем Lasso-регрессию, особенностью который является тот факт, что она зануляет такие маленькие веса. Формула ее функции потерь такова:

$$Q_\alpha(w) = \|y - Xw\|^2 + \alpha\|w\|_1$$

Для улучшения точности модели также будем использовать кросс-валидацию с 3 фолдами. Результаты работы модели приведены ниже на Рис. 21.

```
kf = KFold(n_splits=3)
for train_index, test_index in kf.split(X):
    x_train, x_test = x.iloc[train_index], x.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]

    scaler = StandardScaler()
    x_train = scaler.fit_transform(x_train)
    x_test = scaler.transform(x_test)

    lasso_model = Lasso(alpha = 0.1)
    lasso_model.fit(x_train, y_train)
    y_pred = lasso_model.predict(x_test)
    mae_errors.append(mean_absolute_error(y_pred, y_test))
    r2_errors.append(r2_score(y_test, y_pred))


print('Test mean MAE:', np.mean(mae_errors))
print('Test mean R2-score:', np.mean(r2_errors))
```

```
Test mean MAE: 0.18187148311165546
Test mean R2-score: 0.7666451921525708
```

Рис. 21 Lasso-регрессия с кросс-валидацией

Мы получаем значение R2-score еще немного меньше, чем в Ridge-регрессии, а значение MAE наоборот немного побольше. Это говорит о том, что Lasso-регрессия работает немного хуже, чем Ridge.

Я попыталась также подобрать различные параметры alpha для модели, в надежде, что они смогут улучшить качество ее работы. Однако эти попытки не принесли значительных изменений, поэтому оставлю их за кадром.

Выбор наилучшей модели

Ранее в работе я уже решила использовать логарифмированный таргет для построения моделей. После этого было построено 3 различным модели. Первая – обычная линейная регрессия без регуляризации и с поправкой Уайта. Вторая – ridge-регрессия, в которой используется L2 – регуляризация. Третья – lasso-регрессия, где применяется L1-регуляризация. Качество моделей мы будем сравнивать по результату их работы на тестовой выборке, а именно по значению R2-score. Чем ближе к единице его значение на тестовой выборке – тем лучше модель.

Лучшей моделью оказалась обычная линейная регрессия без регуляризации, но с поправкой Уайта. Она показывает наилучшее качество на тестовой выборке: её значение R2 составляет 0.896, против 0.798 у ridge-регрессии и 0.766 у lasso-регрессии.

Вектор весов нашей модели следующий:

```
[9.33221468e+00, 7.95108630e-04, 7.90405883e-02, 1.16679804e-01, -
1.11351083e-01, 2.04821048e-01, -4.56498554e-02, -4.94207032e-02,
1.29597257e-01, 1.31874872e-01, 3.25062821e-02, -2.35616922e-01,
1.32173869e-01, -2.75288569e-02, 7.55480697e-04, -5.54195824e-02]
```

Уравнение линейной регрессии имеет вид

$$
\begin{aligned}
logprice = \ & 9.3322 + \\
& + 0.0008 * doornumber \\
& + 0.0790 * wheelbase \\
& + 0.1167 * curbweight \\
& - 0.1114 * cylindernumber \\
& + 0.2048 * enginesize \\
& - 0.0456 * \text{boreratio} \\
& - 0.0494 * \text{stroke} \\
& + 0.1296 * \text{compressionratio} \\
& + 0.1319 * \text{horsepower} \\
& + 0.0325 * \text{peakrpm} \\
& - 0.2356 * \text{citympg} \\
& + 0.1322 * \text{highwaympg} \\
& - 0.0275 * \text{fueltype\_diesel} \\
& + 0.0008 * \text{aspiration\_std} \\
& - 0.0554 * \text{enginelocation\_front}
\end{aligned}
$$

Поскольку при построении модели мы маштабировали данные, мы можем трактовать коэффициенты при каждой из переменных как степень их важности в

решении модели. Положительный коэффициент говорит о прямой связи между предиктором и зависимой пременной, отрицательный – об обратной связи.

Мы видим, что самый большой по модулю коэффициент (-0.2356) находится у переменной citympg. Citympg - это количество миль, которое транспортное средство может проехать на одном галлоне топлива при движении в городских условиях. Судя по нашей модели, чем больше проезжает машина, тем дешевле она стоит. Такой результат контринтуитивен, но раз модель приняла такое решение, значит это правда.

Следующий по размеру коэффициент 0.2048 у enginesize. Значит есть прямая связь между размером двигателя и стоимостью автомобиля. Это подтверждает наши предположения, сделанные в начале работы. Также прямая связь есть у цены автомобиля с числом лошадиных сил (0.13) и с весом автомобиля (0.11). Эти факты интуитивно понятны и совпадают с предположенным ранее. Обратная связь есть у цены с переменной fueltype_diesel. Это странно, но это говорит о том, что машины на дизеле стоят немножко дешевле, чем машины на газе.

Заключение

В данной работе мы пытались научиться предсказывать стоимость автомобиля по различным его параметрам. Был проведен первичный анализ данных, отбор переменных для дальнейшего анализа. С помощью корреляционного анализа были исследованы связи между переменными. Далее были построены различные линейные модели (с регуляризацией и без нее), было проведено их сравнение между собой и была выявлена лучшая. Был проведен анализ ошибок модели, тесты на гомо и гетероскедастичность, а также были сделаны выводы о влиянии независимых переменных на целевую.

Выводы по каждому отдельному шагу анализа указаны после вычисления каждого показателя в ходе работы. Если говорить о выводах в целом, из полученных результатов можно судить о том, что на цену автомобиля в основном влияет мощность его двигателя и его размеры. Другие параметры, такие как число дверей, наличие турбонаддува в двигателе, расположение двигателя не оказывают на цену практически никакого влияния.