Отчет о ДЗ №3
Дисциплина: Прикладная мат. статистика
Исполнительница: Смолкина Ю.А
Группа: АДБМ2021

Придумайте разумные цели и методы статистической обработки данных и реализуйте их.

Я выбрала датасет: https://www.kaggle.com/imakash3011/rental-bike-sharing

Инструменты: Python (collab), pandas, numpy, matplotlib.pyplot, seaborn, sklearn.model_selection, train_test_split, StandardScaler и другие

Исходный алгоритм и код: https://colab.research.google.com/drive/13URXYNqdGk5SKxmXRFn900DQH_XTYq2C?usp=sharing

Цели: Изучить датасет (узнать распределения данных и их статистические характеристики). Провести сравнительный анализ данных.Прогнозировать общее количество арендованных велосипедов (произвести обучение и предсказания) оценить результаты по метрикам(см ниже)

Метрики: Распределение данных, корреляция признаков, валидность данных, метрики MSE, R^2, MAPE, регуляризации $L_{1,2}$ и другие

Методы: Предпроцессинг данных, интеллектуальный анализ данных, валидирование данных, методы машинного обучения, Линейные модели (Линейная регрессия, Лассо, Ридж и Эластичная сетка),Нелинейные модели (SVM, random forest, NN) нормализация категориальных признаков, функции потерь,RandomForestRegressor и другие

Исходное описание данных:

Bike sharing systems are new generation of traditional bike rentals where whole process from membership, rental and return back has become automatic. Through these systems, user is able to easily rent a bike from a particular position and return back at another position. Currently, there are about over 500 bike-sharing programs around the world which is composed of over 500 thousands bicycles. Today, there exists great interest in these systems due to their important role in traffic, environmental and health issues.

Apart from interesting real world applications of bike sharing systems, the characteristics of data being generated bythese systems make them attractive for the research. Opposed to other transport services such as bus or subway, the durationof travel, departure and arrival position is explicitly recorded in these systems. This feature turns bike sharing system intoa virtual sensor network that can be used for sensing mobility in the city. Hence, it is expected that most of importantevents in the city could be detected via monitoring these data.

Исходные атрибуты (столбцы):

- instant: record index
- dteday : date
- season : season (1:springer, 2:summer, 3:fall, 4:winter)
- yr : year (0: 2011, 1:2012)
- mnth : month (1 to 12)
- hr : hour (0 to 23)
- holiday : weather day is holiday or not (extracted from http://dchr.dc.gov/page/holiday-schedule)
- weekday : day of the week
- workingday : if day is neither weekend nor holiday is 1, otherwise is 0.
+ weathersit :
    - 1: Clear, Few clouds, Partly cloudy, Partly cloudy
    - 2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
    - 3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds
    - 4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog
- temp : Normalized temperature in Celsius. The values are divided to 41 (max)
- atemp: Normalized feeling temperature in Celsius. The values are divided to 50 (max)
- hum: Normalized humidity. The values are divided to 100 (max)
- windspeed: Normalized wind speed. The values are divided to 67 (max)
- casual: count of casual users
- registered: count of registered users
- cnt: count of total rental bikes including both casual and registered

Посмотрим на наши данные:

	instant	dteday	season	yr	mnth	hr	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
0	1	2011-01-01	1	0	1	0	0	6	0	1	0.24	0.2879	0.81	0.0000	3	13	16
1	2	2011-01-01	1	0	1	1	0	6	0	1	0.22	0.2727	0.80	0.0000	8	32	40
2	3	2011-01-01	1	0	1	2	0	6	0	1	0.22	0.2727	0.80	0.0000	5	27	32
3	4	2011-01-01	1	0	1	3	0	6	0	1	0.24	0.2879	0.75	0.0000	3	10	13
4	5	2011-01-01	1	0	1	4	0	6	0	1	0.24	0.2879	0.75	0.0000	0	1	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
17374	17375	2012-12-31	1	1	12	19	0	1	1	2	0.26	0.2576	0.60	0.1642	11	108	119
17375	17376	2012-12-31	1	1	12	20	0	1	1	2	0.26	0.2576	0.60	0.1642	8	81	89
17376	17377	2012-12-31	1	1	12	21	0	1	1	1	0.26	0.2576	0.60	0.1642	7	83	90
17377	17378	2012-12-31	1	1	12	22	0	1	1	1	0.26	0.2727	0.56	0.1343	13	48	61
17378	17379	2012-12-31	1	1	12	23	0	1	1	1	0.26	0.2727	0.65	0.1343	12	37	49

17379 rows × 17 columns

Исходный размер: 17379 rows × 17 columns

I. Data Preprocessing

Удалим колонку instant (This column corresponds to the observation number and does not make sense)

Проведем проверку данных на валидность:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17379 entries, 0 to 17378
Data columns (total 16 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   dteday      17379 non-null  object 
 1   season      17379 non-null  int64  
 2   yr          17379 non-null  int64  
 3   mnth        17379 non-null  int64  
 4   hr          17379 non-null  int64  
 5   holiday     17379 non-null  int64  
 6   weekday     17379 non-null  int64  
 7   workingday  17379 non-null  int64  
 8   weathersit  17379 non-null  int64  
 9   temp        17379 non-null  float64
 10  atemp       17379 non-null  float64
 11  hum         17379 non-null  float64
 12  windspeed   17379 non-null  float64
 13  casual      17379 non-null  int64  
 14  registered  17379 non-null  int64  
 15  cnt         17379 non-null  int64  
dtypes: float64(4), int64(11), object(1)
memory usage: 2.1+ MB
```

И давайте посмотрим на статистические значения (разброс данных) в каждом столбце

	season	yr	mnth	hr	holiday	weekday	workingday	weathersit	temp	atemp
count	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000
mean	2.501640	0.502561	6.537775	11.546752	0.028770	3.003683	0.682721	1.425283	0.496987	0.475775
std	1.106918	0.500008	3.438776	6.914405	0.167165	2.005771	0.465431	0.639357	0.192556	0.171850
min	1.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.020000	0.000000
25%	2.000000	0.000000	4.000000	6.000000	0.000000	1.000000	0.000000	1.000000	0.340000	0.333300
50%	3.000000	1.000000	7.000000	12.000000	0.000000	3.000000	1.000000	1.000000	0.500000	0.484800
75%	3.000000	1.000000	10.000000	18.000000	0.000000	5.000000	1.000000	2.000000	0.660000	0.621200
max	4.000000	1.000000	12.000000	23.000000	1.000000	6.000000	1.000000	4.000000	1.000000	1.000000

*Представлены не все колонки

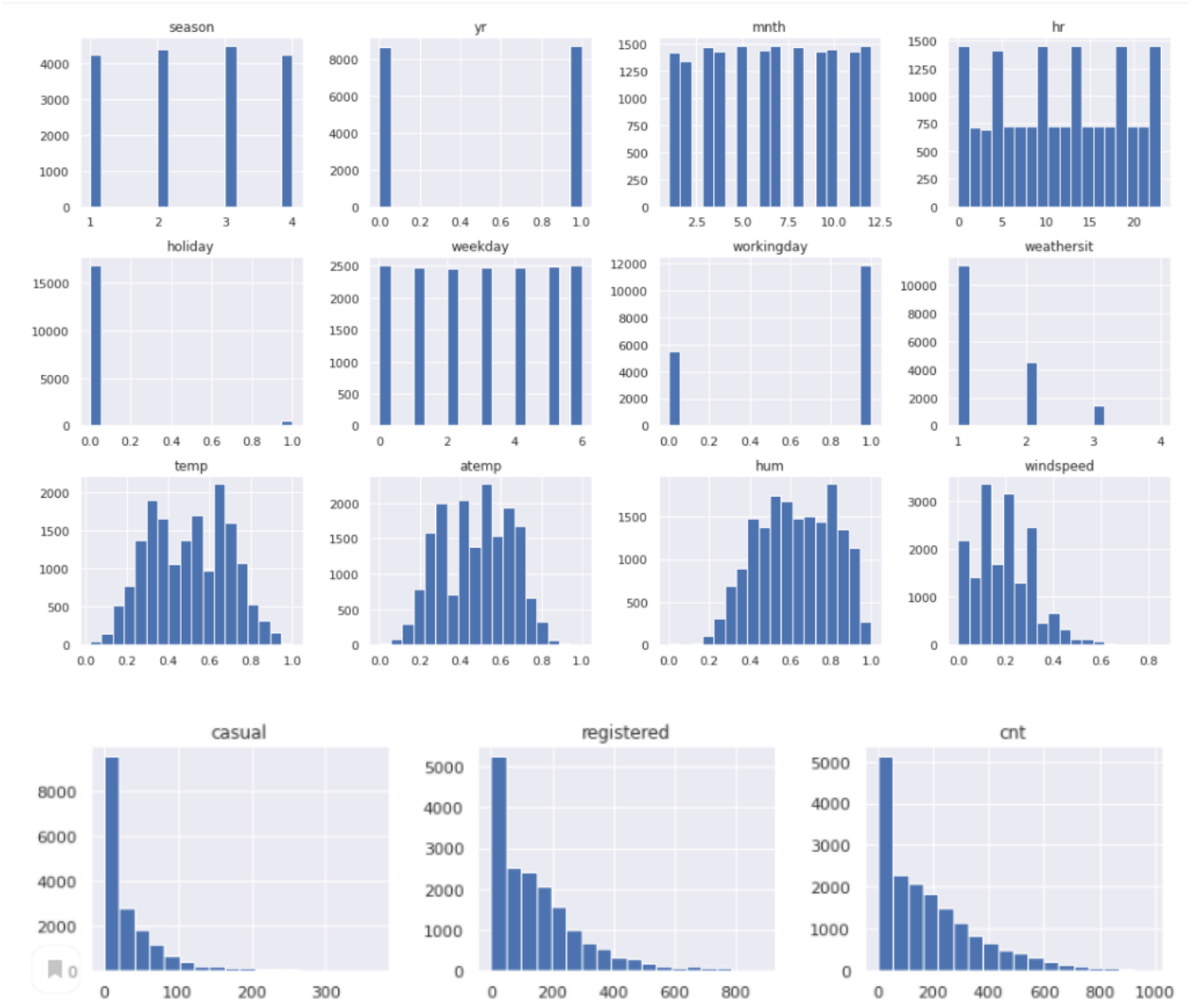
Сразу заметим, что Минимальное значение арендованных велосипедов равно 1. Можно предположить, что информация об аренде не записывалась, если велосипеды не были арендованы.

Давайте определим как распределены данные и как они коррелированы:

Мы можем сделать несколько диаграмм для визуализации ваших данных:

Гистограммы для описания распределения каждой функции

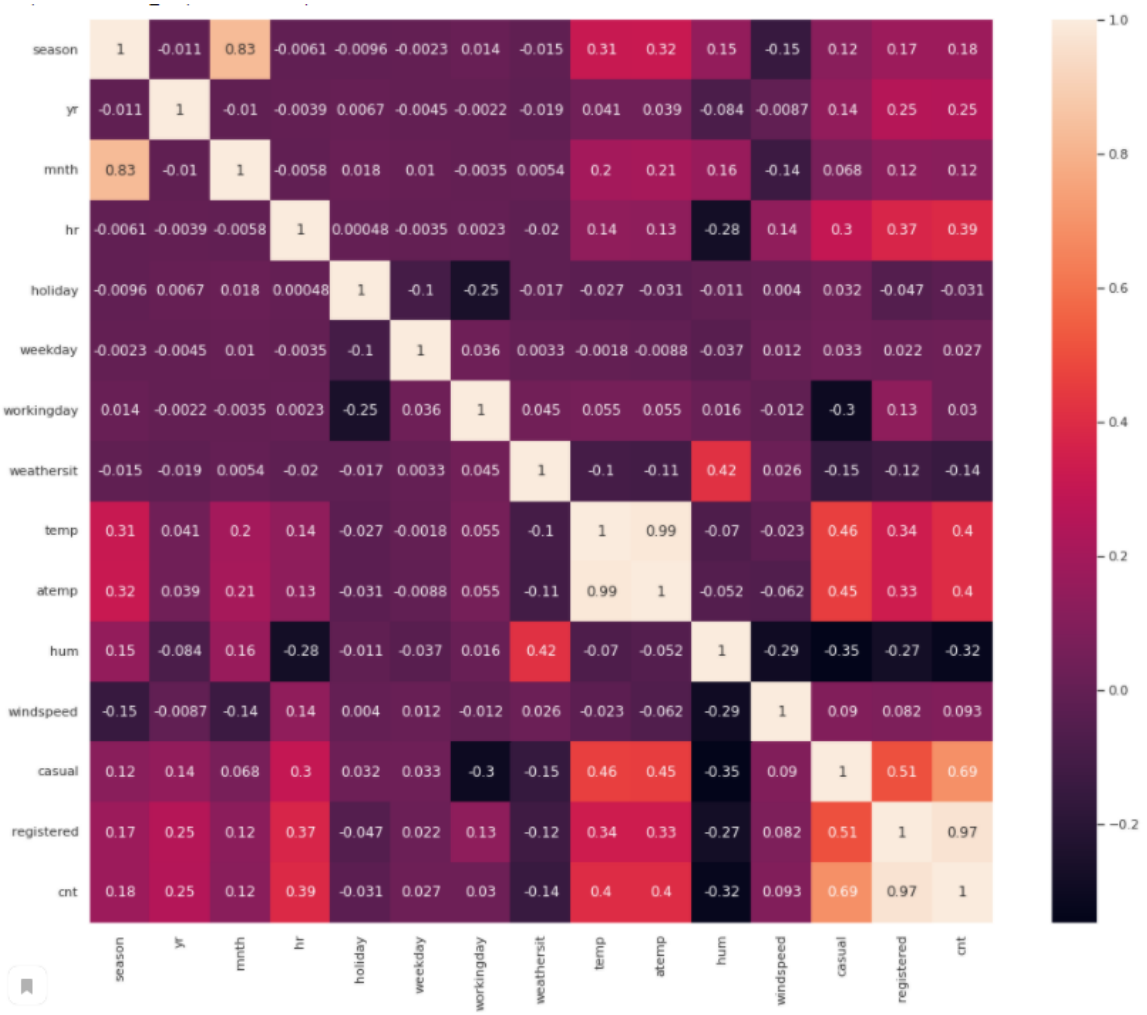
Матрица тепловой карты для оценки корреляции между каждой парой столбцов



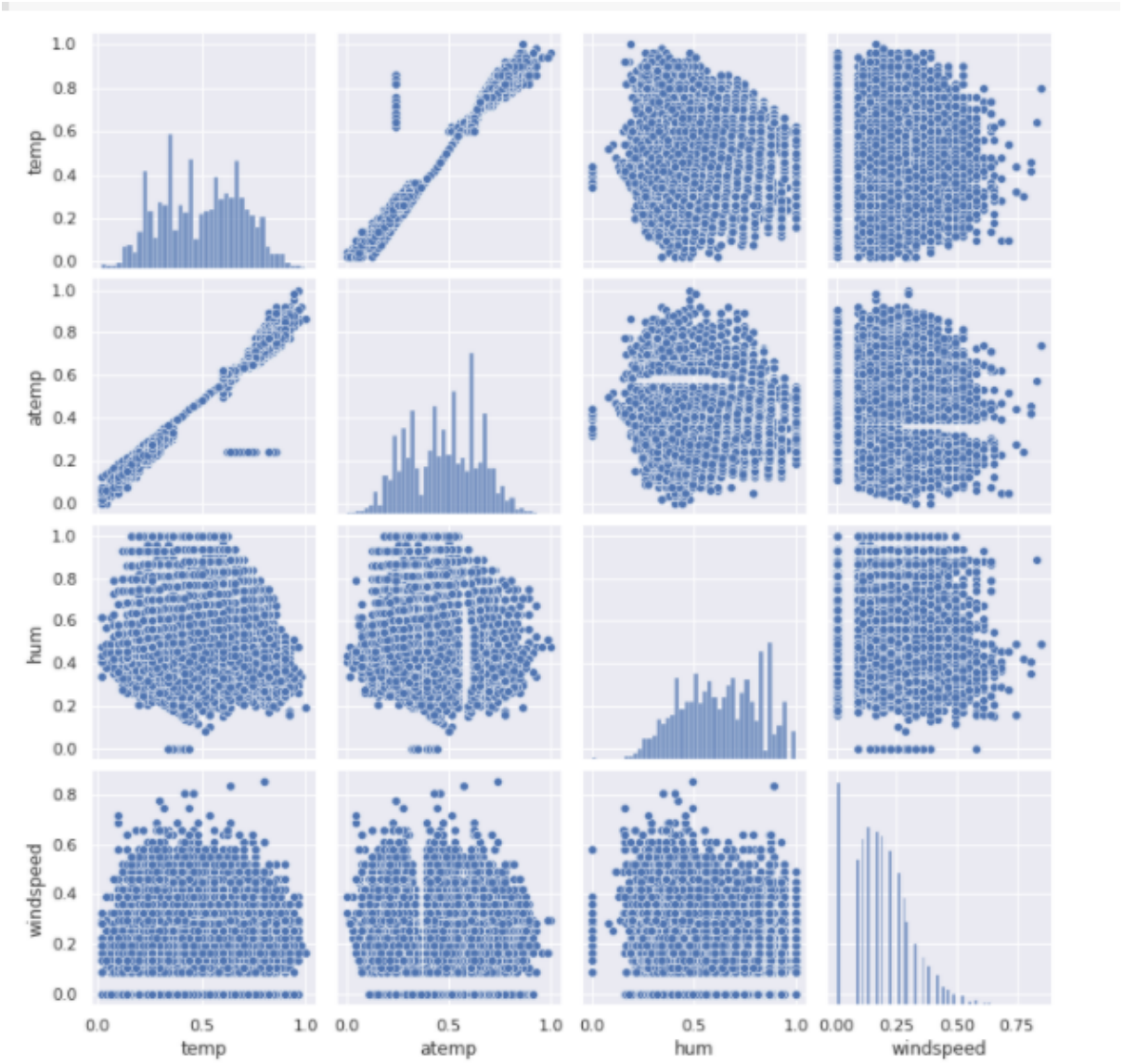
Глядя на эти гистограммы, мы можем отметить некоторые моменты:

- Признаки 'season', 'mnth', 'yr', 'dayday' имеют дискретное равномерное распределение, что означает, что наши данные сбалансированы «по времени». Данные содержат практически одинаковое количество записей для разных сезонов, лет, дней недели.
- Распределение признака 'hr' не выглядит равномерным, что может быть связано с сделанным ранее предположением: если в этот час велосипеды не берутся напрокат, то запись информации не ведется.
- Немного странно выглядит информация о рабочем дне, т.к. рабочие дни в обычном понимании это 5 дней из семи, но соотношение высот столбиков больше похоже на соотношение 1:2. Праздники также считаются нерабочими днями, но на гистограмме «Праздники» мы видим, что таких дней немного.
- Распределения признаков temp, atemp и hum (влажность) близки к норме; полимодальность можно увидеть в распределениях характеристик «temp» и «atemp»; положительная асимметрия может наблюдаться в распределении признаков «скорости ветра».
- Целевой признак «cnt», по-видимому, имеет экспоненциальное распределение (а также «случайное» и «зарегистрированное») или может быть распределением Пуассона с $\lambda = 1$.

Матрица корреляции признаков:



- Как мы видим, функции «temp» и «atemp» сильно коррелированы. Это вполне ожидаемо, потому что temp - нормализованная температура в градусах Цельсия, а atemp - нормированная температура ощущения в градусах Цельсия.
- Также очевидна высокая корреляция между «сезоном» и «месяцем». Информация об обеих функциях может быть избыточной.
- «cnt» представляет собой сумму «случайного» и «зарегистрированного», таким образом, этот признак соотносится с ними обоими; но это больше коррелирует с «зарегистрированным». Причина может заключаться в том, что зарегистрированных пользователей больше, чем случайных, и они вносят наибольший вклад в общую сумму и определяют ее поведение. Далее мы немного остановимся на этом вопросе.



Как мы видим, между функциями temp и atemp есть зависимость, поэтому одну из них мы можем опустить. Также мы можем видеть странный пик на «нуле» для функции «скорость ветра», поэтому мы можем попытаться исключить эти наблюдения из выборки. Но есть много наблюдений со значением 0 по этому признаку, и они не выглядят как выбросы, они, вероятно, отражают совершенно безветренную погоду.

Изобразим, как изменяется среднемесячное количество велосипедов напрокат в течение года. Нарисуем кривые как для 2011, так и для 2012 года.

Как видим, по каждому месяцу среднемесячное количество арендованных велосипедов в 2012 году больше, чем в 2011 году. Это может быть связано с ростом популярности велосипедов, с улучшением работы системы проката и так далее. Можно сделать вывод, что признак важен, потому что есть разница в два года. Столбец «год» не следует удалять из матрицы признаков.

Однако следует отметить, что для реальной проблемы есть некоторые нюансы: в отдельные годы количество пользователей может выйти на плато и перестать расти из года в год или, наоборот, пользование велопрокатом может упасть. по какой-то причине (изоляция из-за коронавирусной инфекции, например). Но в нашем кейсе мы используем данные только об этих двух годах и не будем вдаваться в такие нюансы.

Вот мы и подобрались к нашей задаче:

Общее количество арендованных велосипедов включает случайных пользователей и зарегистрированных пользователей. Таким образом, столбец «cnt» представляет собой сумму «случайных» и «скорректированных» столбцов. Таким образом, мы можем предсказать как «случайные», так и «подстроенные» отдельно, а затем рассчитать общее количество арендованных велосипедов. Но для упрощения задачи, так как она образовательная, мы будем прогнозировать сразу общее количество арендованных велосипедов.

Модификация данных:

*Тк далее я буду использовать методы ML, то для простоты и наглядности разделю некоторые столбцы. Например, добавим информацию о том, за какой день месяца сделана запись. Преобразуем признак "день" в признак, имеющий три ранга: 1 ранг - первые 10 дней месяца, 2 ранг - с 11 по 20 число месяца, 3 ранг - 21 число месяца и позже. (бинарные или почти бинарные атрибуты более удобны в использовании, однако есть свои минусы как “раздавание весов” - в данном случае, я конечно же не имею намерения ранжировать дни месяца с весовыми коэффициентами, все дни одинаковы в плане полезности и интереса для исследований)

У нас есть много категориальных функций, которые мы должны кодировать в фиктивные(dummy) переменные. Было решено отказаться от функции «будний день», потому что функции «рабочий день» может быть достаточно. Кроме того, на мой взгляд, информации о месяце достаточно, поэтому столбец «сезон» тоже можно опустить (эти признаки сильно коррелированы).

Теперь нам нужно удалить избыточные переменные. Если мы знаем, что в столбцах mnth2, mnth3, ..., mnth12 есть нули, значит, в mnth 1 столбце есть 1. Таким образом, n-1 признаков полностью кодируют n возможных вариантов исходного категориального признака.

II. Building regression models and tuning their parameters

- я разбила данные на обучающую и тестовую выборку в соотношении 80\20

Наши данные представлены либо уже предварительно нормализованными признаками (температура, влажность, скорость ветра), либо категориальными закодированными признаками.

Насколько я знаю, масштабирование рекомендуется почти всегда, так как оно никогда не бывает лишним.

Для каждой модели будут оцениваться показатели производительности R^2(**Коэффициент детерминации**) и RMSE (**Среднеквадратичное отклонение).** Также будет создан график «Прогнозируемое и наблюдаемое значения» для целевого объекта (признака).

Теоретическая справка:

R^2 - это доля дисперсии зависимой переменной, объясняемая рассматриваемой моделью зависимости, то есть объясняющими переменными. Более точно — это единица минус доля необъяснённой дисперсии (дисперсии случайной ошибки модели, или условной по факторам дисперсии зависимой переменной) в дисперсии зависимой переменной. Его рассматривают как универсальную меру зависимости одной случайной величины от множества других.

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

$$R^2 = 1 - \frac{D[y|x]}{D[y]} = 1 - \frac{\sigma^2}{\sigma_y^2},$$

где $D[y] = \sigma_y^2$ — дисперсия случайной величины *y*, а $D[y|x] = \sigma^2$ — условная (по факторам *x*) дисперсия зависимой переменной (дисперсия ошибки модели).

В данном определении используются истинные параметры, характеризующие распределение случайных величин. Если использовать выборочную оценку значений соответствующих дисперсий, то получим формулу для выборочного коэффициента детерминации (который обычно и подразумевается под коэффициентом детерминации):

$$R^2 = 1 - \frac{\hat{\sigma}^2}{\hat{\sigma}_y^2} = 1 - \frac{SS_{res}/n}{SS_{tot}/n} = 1 - \frac{SS_{res}}{SS_{tot}},$$

где $SS_{res} = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$ — сумма квадратов остатков регрессии, $y_i, \hat{y}_i$ — фактические и расчётные значения объясняемой переменной.

$SS_{tot} = \sum_{i=1}^{n} (y_i - \bar{y})^2 = n\hat{\sigma}_y^2$ — общая сумма квадратов.

$$\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i$$

RMSE - является часто используемой мерой различий между значениями (выборочными или популяционными значениями), предсказанными моделью или оценщиком, и наблюдаемыми значениями. RMSD представляет квадратный корень из второго выборочного момента разностей между прогнозируемыми и наблюдаемыми значениями или среднее квадратичное этих разностей. Эти отклонения называются *невязками*, когда вычисления выполняются по выборке данных, которая использовалась для оценки, и называются *ошибками* (или ошибки прогнозирования) при вычислении вне выборки.

Формула

RMSD оценщика $\hat{\theta}$ относительно оцениваемого параметра θопределяется как квадратный корень из среднеквадратичной ошибки:

$$\mathrm{RMSD}(\hat{\theta}) = \sqrt{\mathrm{MSE}(\hat{\theta})} = \sqrt{\mathrm{E}((\hat{\theta} - \theta)^2)}.$$

Для несмещенной оценкиRMSD-это квадратный корень из дисперсии, известный как стандартное отклонение.

RMSD прогнозируемых значений $\hat{y}_t$для времен *t* зависимой переменной регрессии y_t, с переменными, наблюдаемыми в течение *T* раз, вычисляется для *T* различных прогнозов как квадратный корень из среднего значения квадратов отклонений:

$$\mathrm{RMSD} = \sqrt{\frac{\sum_{t=1}^{T} (\hat{y}_t - y_t)^2}{T}}.$$

(Для регрессий по данным поперечного сеченияиндекс *t* заменяется на *i*, а *T*-на *n*.)

В некоторых дисциплинах RMSD используется для сравнения различий между двумя вещами, которые могут меняться, ни одна из которых не принимается в качестве "стандарта". Например, при измерении средней разницы между двумя временными рядами $x_{1,t}$и $x_{2,t}$формула становится

$$\mathrm{RMSD} = \sqrt{\frac{\sum_{t=1}^{T} (x_{1,t} - x_{2,t})^2}{T}}.$$

1) Linear regression

Регрессионная модель

$$y = f(x, b) + \varepsilon,\ E(\varepsilon),$$

где b — параметры модели, ε — случайная ошибка модели; называется линейной регрессией, если функция регрессии $f(x, b)$ имеет вид

$$f(x, b) = b_0 + b_1 x_1 + b_2 x_2 + \ldots + b_k x_k,$$

где b_j — параметры (коэффициенты) регрессии, x_j — регрессоры (факторы модели), k — количество факторов модели[1].

Коэффициенты линейной регрессии показывают скорость изменения зависимой переменной по данному фактору, при фиксированных остальных факторах (в линейной модели эта скорость постоянна):

$$\forall j \quad b_j = \frac{\partial f}{\partial x_j} = const$$

Параметр b_0, при котором нет факторов, называют часто *константой*. Формально — это значение функции при нулевом значении всех факторов. Для аналитических целей удобно считать, что константа — это параметр при «факторе», равном 1 (или другой произвольной постоянной, поэтому константой называют также и этот «фактор»). В таком случае, если перенумеровать факторы и параметры исходной модели с учетом этого (оставив обозначение общего количества факторов — k), то линейную функцию регрессии можно записать в следующем виде, формально не содержащем константу:

$$f(x, b) = b_1 x_1 + b_2 x_2 + \ldots + b_k x_k = \sum_{j=1}^{k} b_j x_j = x^T b,$$

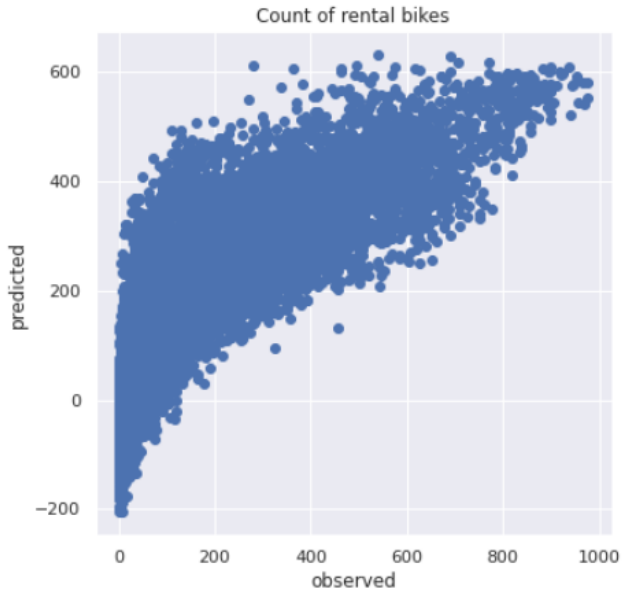
где $x^T = (x_1, x_2, \ldots, x_k)$ — вектор регрессоров, $b = (b_1, b_2, \ldots, b_k)^T$ — вектор-столбец параметров (коэффициентов).

Линейная модель может быть как с константой, так и без константы. Тогда в этом представлении первый фактор либо равен единице, либо является обычным фактором соответственно.

Loss-function: $L(f, X, y) = \sum_{i=1}^{N} (y_i - \langle x_i, w \rangle)^2$

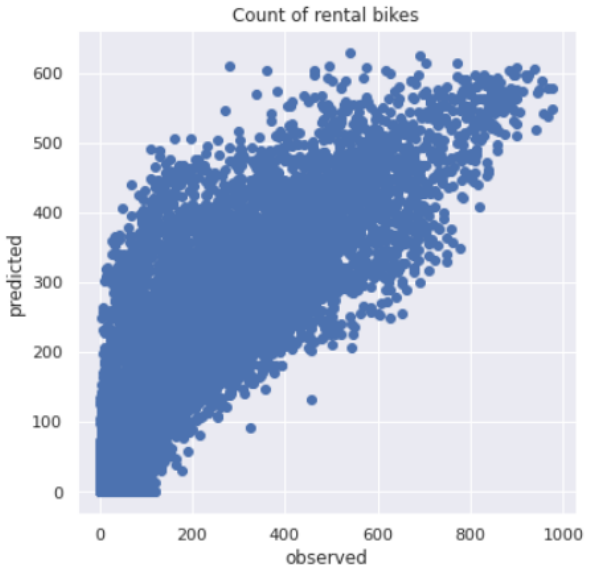
В математической оптимизации и теории принятия решений , функция потерь или функция затрат (иногда также называется функция ошибки) является функцией , которая отображает события или значения одного или несколько переменных на вещественное число интуитивно , представляющее некоторые «стоимость» , связанную с событием. Задача оптимизации стремится к минимизации функции потерь. Целевая функция является либо функцией потерь или его противоположность (в определенных областях, по- разному называется функция вознаграждения , а функция прибыли , а функция полезности , в функции пригодности и т.д.), в этом случае он должен быть максимальными.

```
R2-score for train : 0.6846813036685602
RMSE for train : 102.06141362675486
```



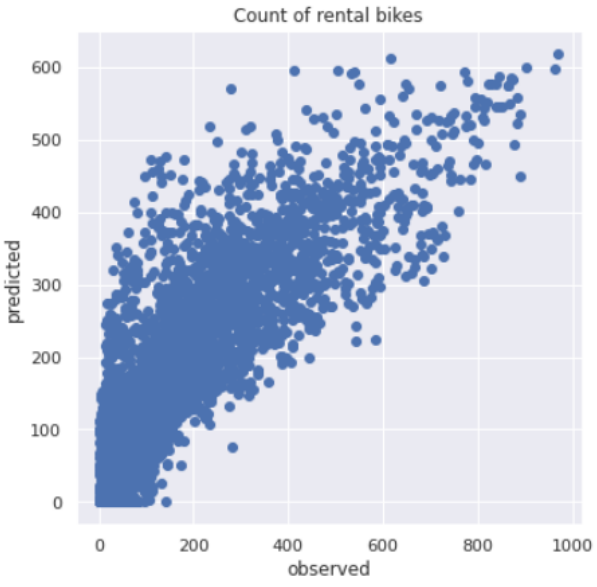
Мы можем заметить, что некоторые значения целевой переменной принимают отрицательные значения, что противоречит здравому смыслу. Мы можем «уточнить» данные, предсказанные моделью, присвоив значение 0 целевой переменной, если модель предсказывает его как отрицательное. Кроме того, модели предсказывают реальные значения, а значение принадлежит множеству натуральных чисел. Мы могли бы преобразовать функцию следующим образом, чтобы модель выдавала адекватные результаты для заказчика, но я не уверен, что именно так мы и поступаем на практике. Это дополнительный исследовательский вопрос.

```
R2-score for train : 0.7023974519721736
RMSE for train : 99.1528132285863
```



MAPE-score for train : 1.3000852214089726

```
R2-score for test : 0.6891140827565831
RMSE for test : 100.2452989694657
```



MAPE-score for test : 1.406520585370354

```
best_results
```

	R2_train	RMSE_train	R2_test	RMSE_test	Best parameters
Linear regression	0.702	99.153	0.689	100.245	-

2) Ridge regression

In this case the loss function is the linear least squares function and regularization is given by the L_2-norm.

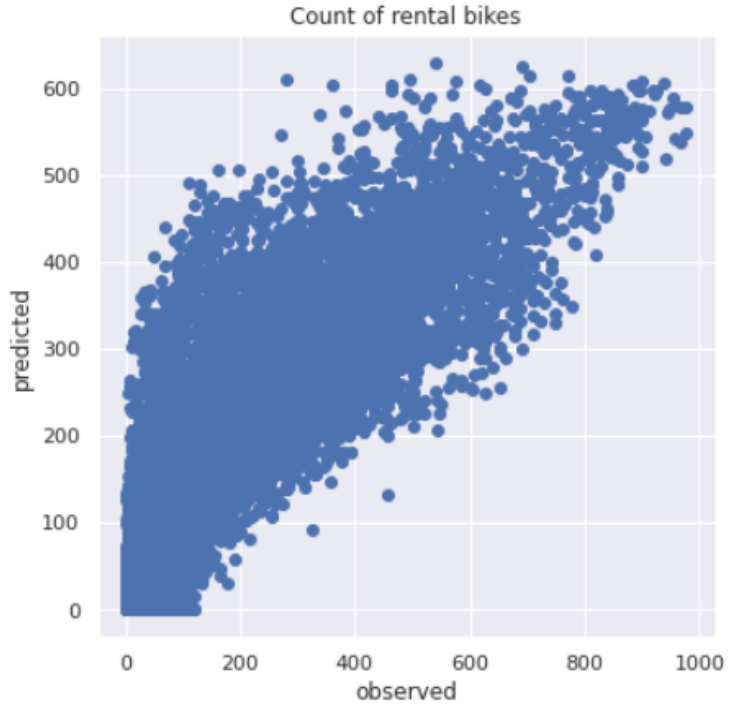
Loss-function: $L(f, X, y) = \sum_{i=1}^{N}(y_i - \langle x_i, w \rangle)^2 + \alpha_{II} \sum_{j=1} w_j^2$

L_2 regularization helps to solve the problem of multicollinearity by reducing weights for linearly dependent features.

Без специальной настройки параметров

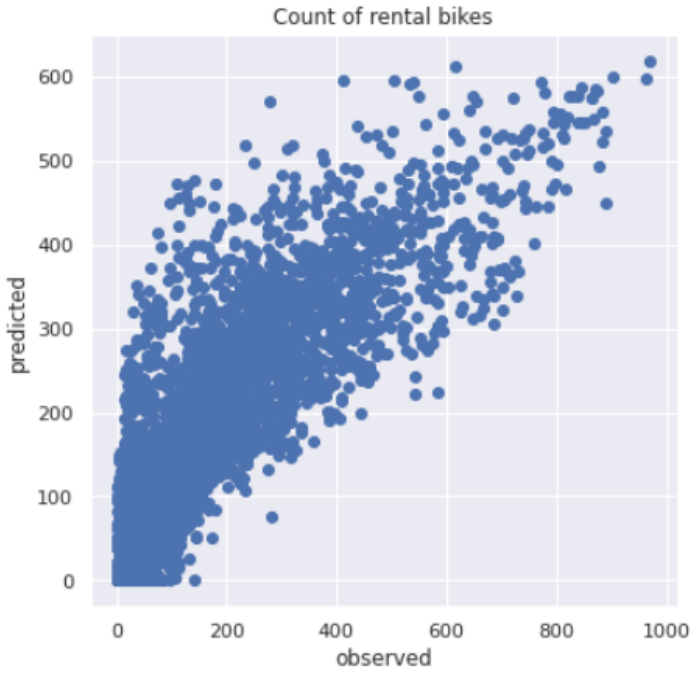
MAPE-score for train : 1.3

```
R2-score for train : 0.7023848432346003
RMSE for train : 99.15491364501132
```



```
MAPE-score for test : 1.4
```

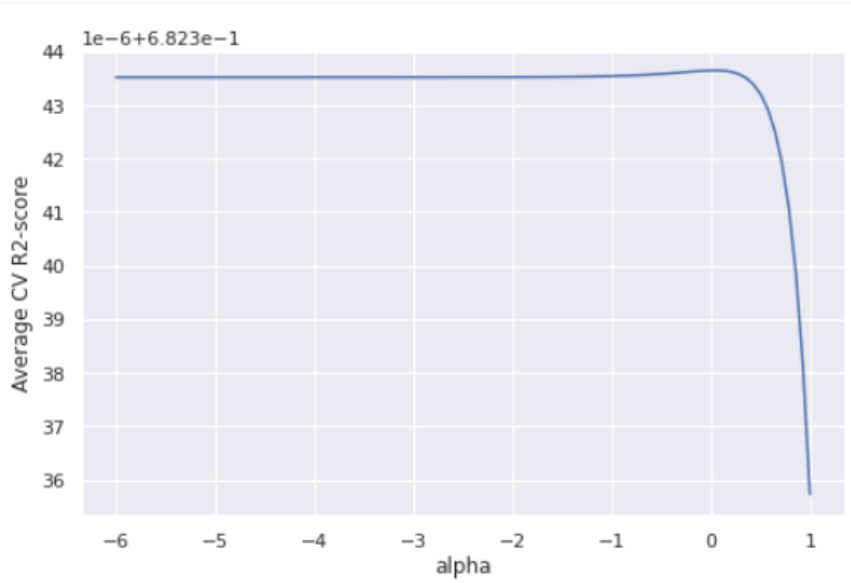
```
R2-score for test : 0.6891275663573069
RMSE for test : 100.24312504944064
```



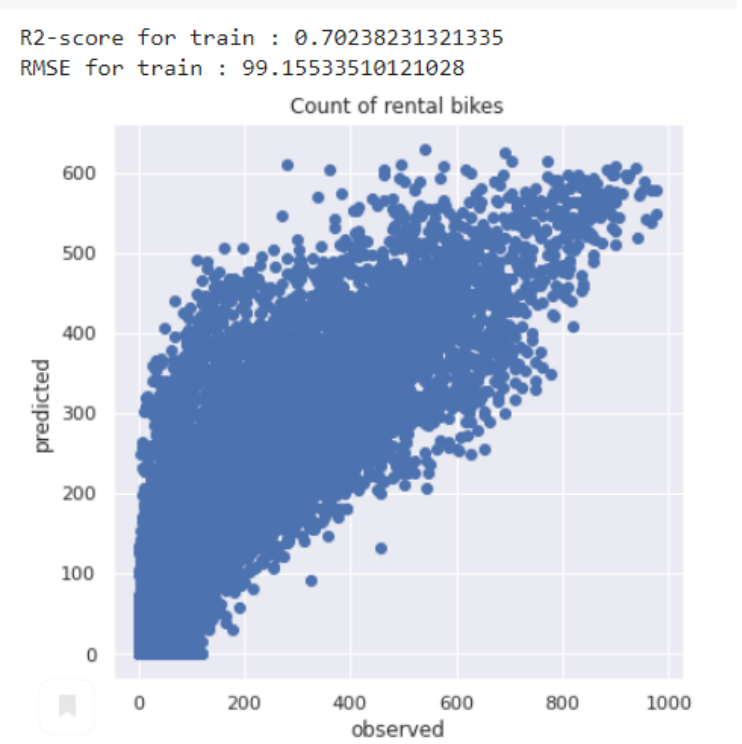
L_2 регуляризация настроек параметров (regularization parameter tuning):

```
Best params: {'alpha': 1.0235310218990248}
Best cross validaton score 0.6823436386615556
```

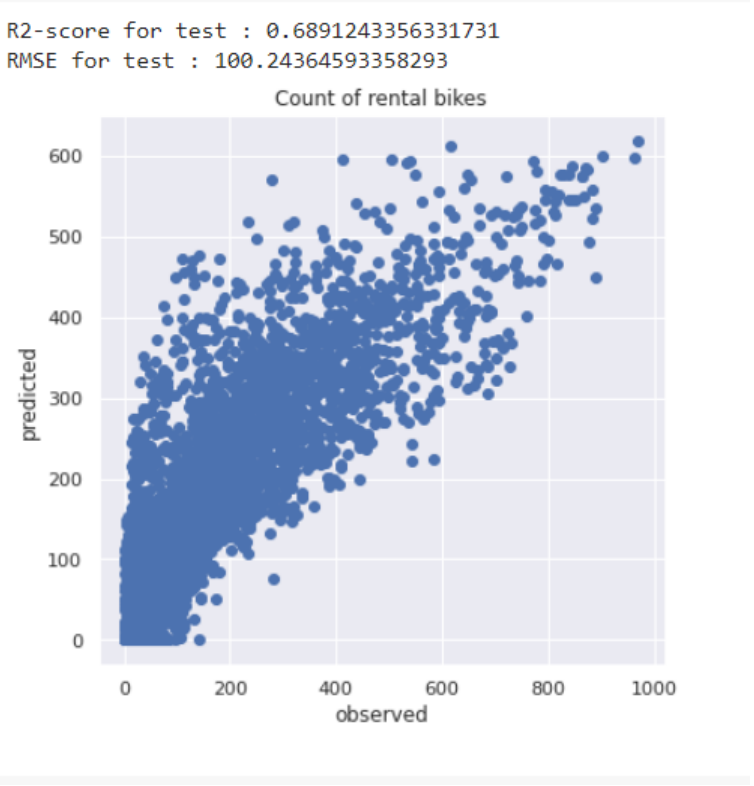
Зависимость усредненного по складкам коэффициента детерминации от гипер параметра:



Фитинг с лучшими параметрами показал результаты :
MAPE-score for train : 1.30



Проверка модели и параметров на тестовой выборке:
MAPE-score for test : 1.40



Промежуточные результаты:

	R2_train	RMSE_train	R2_test	RMSE_test	R2_train (after tuning)	RMSE_train (after tuning)	R2_test (after tuning)	RMSE_test (after tuning)
Linear regression	0.702	99.153	0.689	100.245				
Ridge regression	0.702	99.155	0.689	100.243	0.702	99.155	0.689	100.244

Лучшие результаты:

	R2_train	RMSE_train	R2_test	RMSE_test	Best parameters
Linear regression	0.702	99.153	0.689	100.245	-
Ridge regression	0.702	99.155	0.689	100.243	alpha_l = 1.024

3) Lasso

В этом случае функция потерь является линейной функцией наименьших квадратов, а регуляризация задается нормой L_1

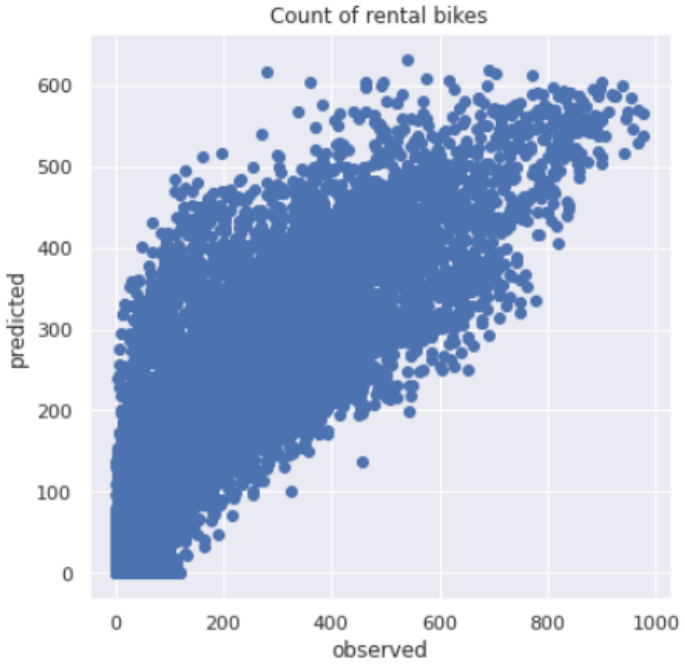
$$\text{Loss-function: } L(f, X, y) = \sum_{i=1}^{N}(y_i - \langle x_i, w\rangle)^2 + \alpha_I \sum_{j=1} |w_j|$$

Без калибровки параметров:

MAPE-score for train : 1.371

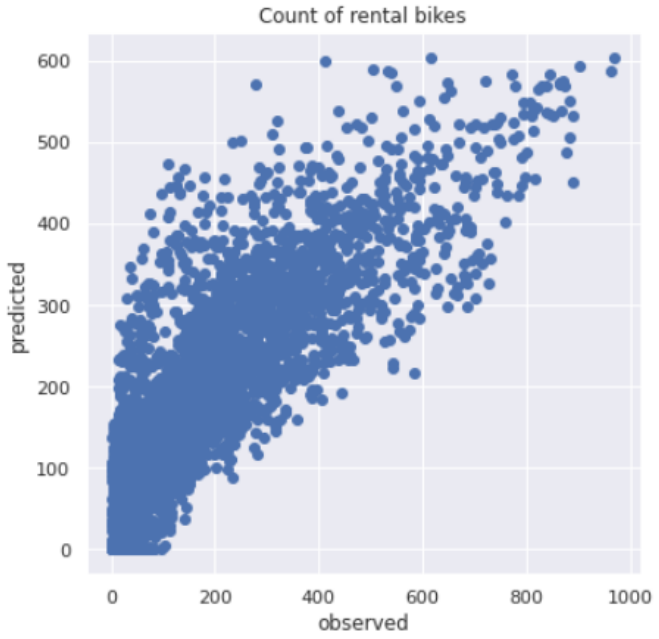
R2-score for train : 0.6930259048933528

RMSE for train : 100.7018808428811



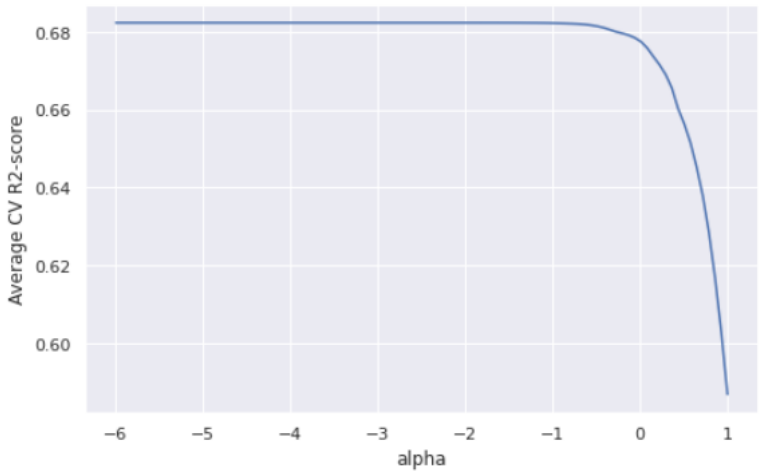
```
MAPE-score for test : 1.458
```

```
R2-score for test : 0.6807947367103828
RMSE for test : 101.57773217190652
```



L_1 regularization parameter tuning:

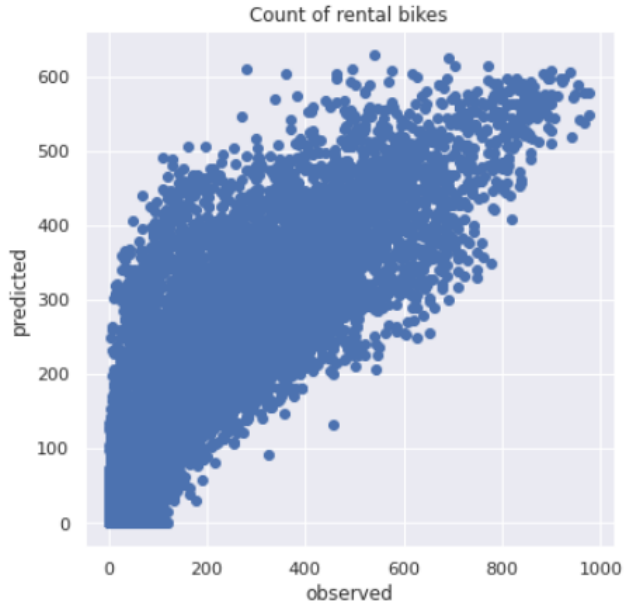
```
Best params: {'alpha': 0.005590810182512223}
Best cross validaton score 0.682343888774488
```



Используем лучшие параметры:

MAPE-score for train : 1.300

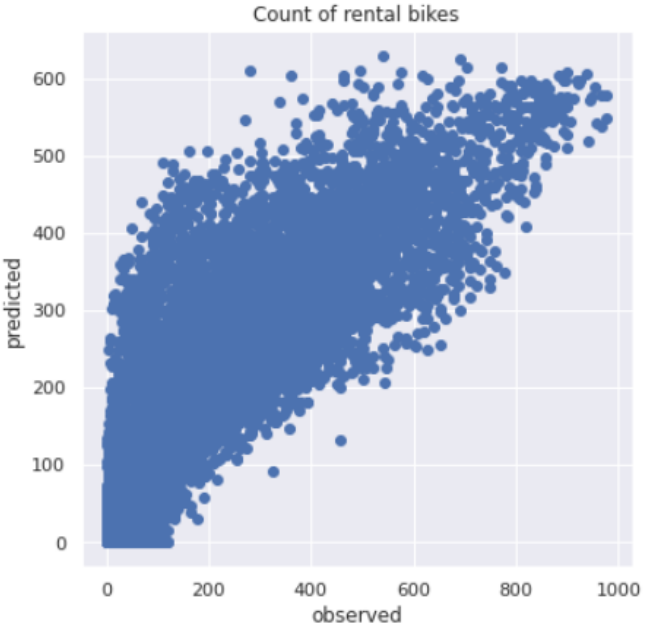
```
R2-score for train : 0.7023451510767754
RMSE for train : 99.16152544077876
```



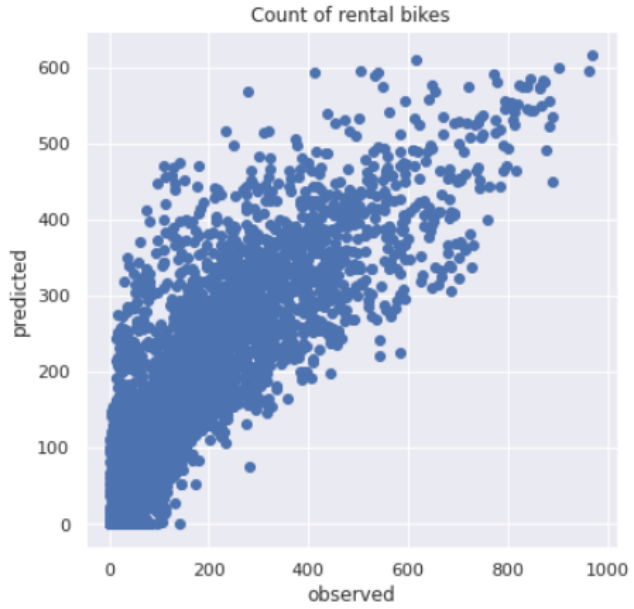
Протестируем:

MAPE-score for test : 1.406

```
R2-score for train : 0.7023451510767754
RMSE for train : 99.16152544077876
```



```
R2-score for test : 0.6890983207388397
RMSE for test : 100.24784017198797
```



Текущие значения:

	R2_train	RMSE_train	R2_test	RMSE_test	R2_train (after tuning)	RMSE_train (after tuning)	R2_test (after tuning)	RMSE_test (after tuning)
Linear regression	0.702	99.153	0.689	100.245				
Ridge regression	0.702	99.155	0.689	100.243	0.702	99.155	0.689	100.244
Lasso	0.693	100.702	0.681	101.578	0.702	99.162	0.689	100.248

```
index = [ Lasso ]))
best_results
```

	R2_train	RMSE_train	R2_test	RMSE_test	Best parameters
Linear regression	0.702	99.153	0.689	100.245	-
Ridge regression	0.702	99.155	0.689	100.243	alpha_I = 1.024
Lasso	0.702	99.162	0.689	100.248	alpha_II = 0.006

4) Elastic Net

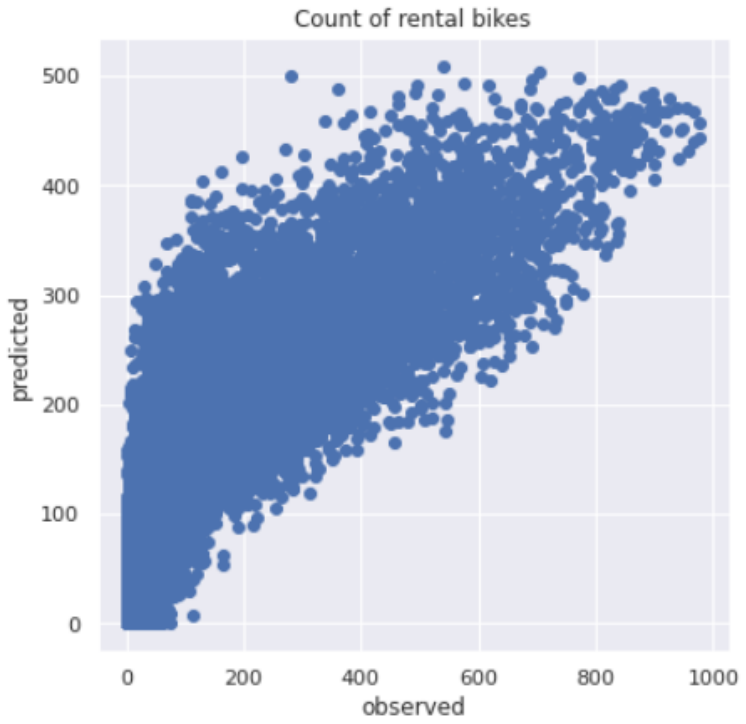
В этом случае функция потерь является линейной функцией наименьших квадратов, а регуляризация задается как L2-нормой, так и L1-нормой.

Loss-function: $L(f, X, y) = \sum_{i=1}^{N}(y_i - \langle x_i, w \rangle)^2 + \alpha_I \sum_{j=1} |w_j| + \alpha_{II} \sum_{j=1} w_j^2$

Без настройки параметров:

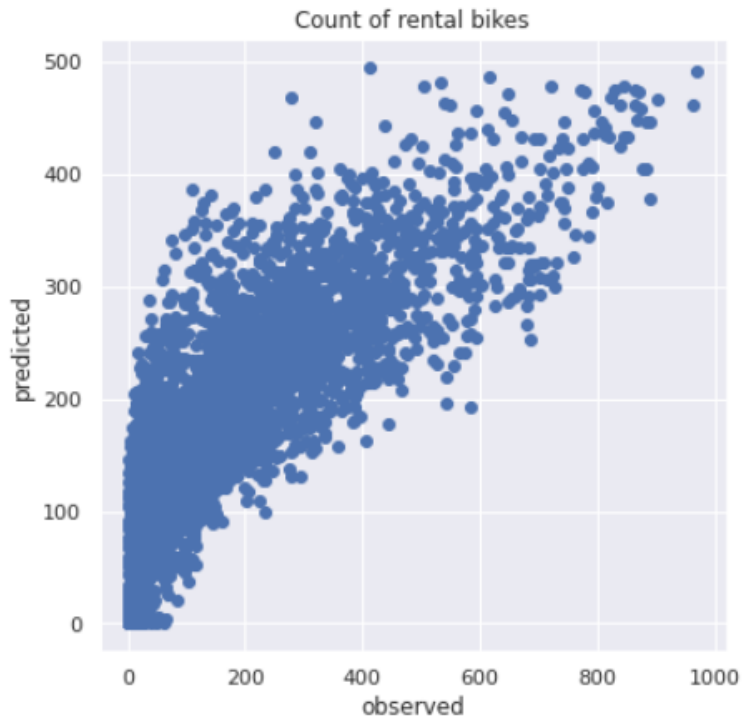
MAPE-score for train : 2.252

R2-score for train : 0.6034911722924832
RMSE for train : 114.44930204670189



MAPE-score for test : 2.36

R2-score for test : 0.5975970398167321
RMSE for test : 114.04968079437155



Параметры настройки. Elastic Net в sklearn минимизирует целевую функцию:

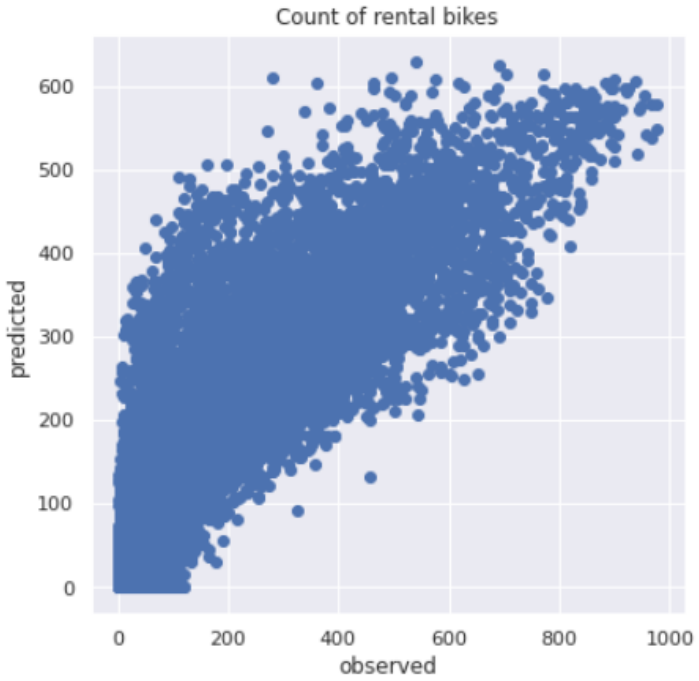
$$\frac{1}{2 \cdot N} \cdot \sum_{i=1}^{N}(y_i - \langle x_i, w \rangle)^2 + \alpha \cdot L1ratio \cdot \sum_{j=1} |w_j| + 0.5 \cdot \alpha \cdot (1 - L1ratio) \cdot \sum_{j=1} w_j^2$$

And tuning parameters are α and $L1\_ratio$.

```
Best params: {'alpha': 0.01, 'l1_ratio': 0.99}
Best cross validaton score 0.6823430479014566
```

```
MAPE-score for train : 1.300
```

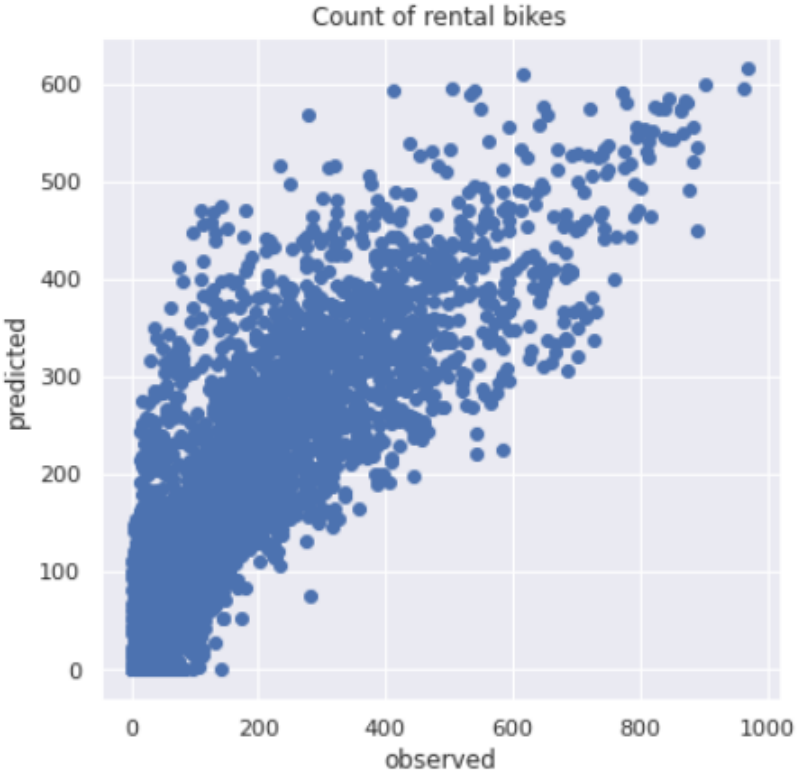
```
R2-score for train : 0.702309645864617
RMSE for train : 99.1674394146753
```



Теперь проверим все на тестовой выборке:

```
MAPE-score for test : 1.407
```

```
R2-score for test : 0.689064562786665
RMSE for test : 100.25328251945845
```



Текущие значения:

	R2_train	RMSE_train	R2_test	RMSE_test	R2_train (after tuning)	RMSE_train (after tuning)	R2_test (after tuning)	RMSE_test (after tuning)
Linear regression	0.702	99.153	0.689	100.245				
Ridge regression	0.702	99.155	0.689	100.243	0.702	99.155	0.689	100.244
Lasso	0.693	100.702	0.681	101.578	0.702	99.162	0.689	100.248
ElasticNet	0.603	114.449	0.598	114.050	0.702	99.167	0.689	100.253

```
best_results
```

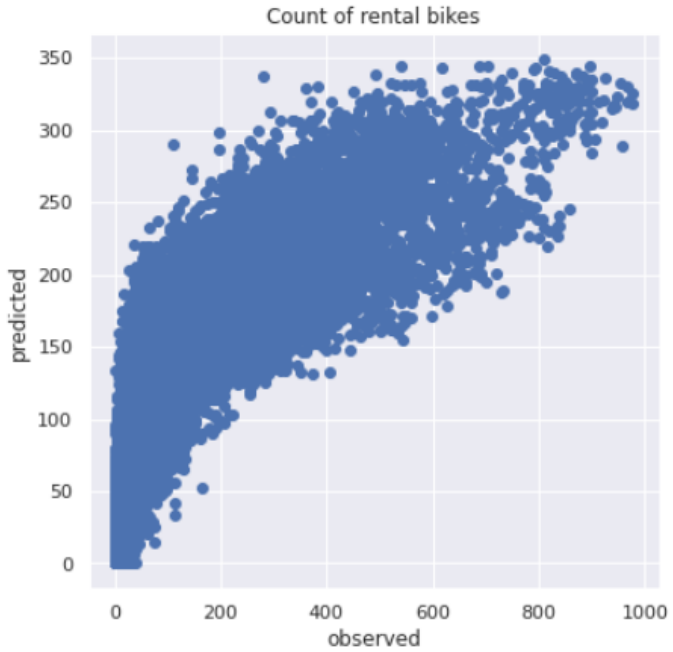
	R2_train	RMSE_train	R2_test	RMSE_test	Best parameters
Linear regression	0.702	99.153	0.689	100.245	-
Ridge regression	0.702	99.155	0.689	100.243	alpha_l = 1.024
Lasso	0.702	99.162	0.689	100.248	alpha_ll = 0.006
ElasticNet	0.702	99.167	0.689	100.253	alpha = 0.01, l1_ratio = 0.99

5) SVM

Из документации:
Эпсилон-регрессия опорных векторов.
Свободными параметрами в модели являются C и эпсилон.
Реализация основана на libsvm. Сложность времени подгонки более чем квадратична по количеству выборок, что затрудняет масштабирование до наборов данных с более чем парой 10000 выборок.

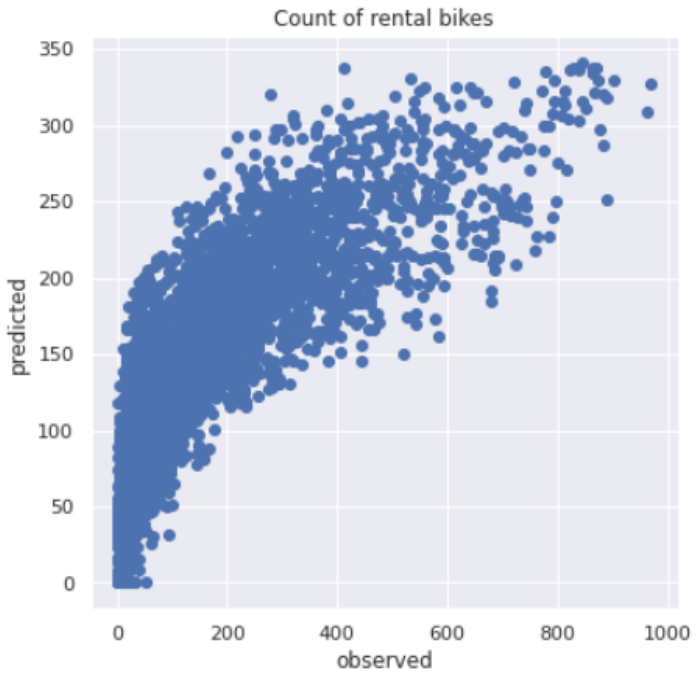
Без настройки параметров:
MAPE-score for train : 1.45

R2-score for train : 0.46527836053983496
RMSE for train : 132.90783941634604



```
MAPE-score for test : 1.555
```

```
R2-score for test : 0.4739857608302932
RMSE for test : 130.39539101401746
```



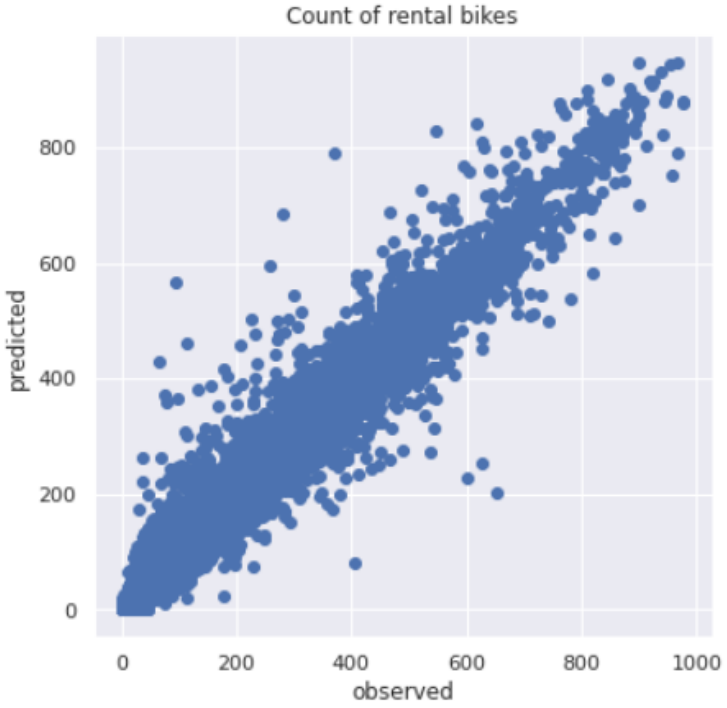
С настройкой параметров:
С — параметр регуляризации. Сила регуляризации обратно пропорциональна С. Должна быть строго положительной. Штраф представляет собой штраф в квадрате l2.

kernel указывает тип ядра, который будет использоваться в алгоритме.

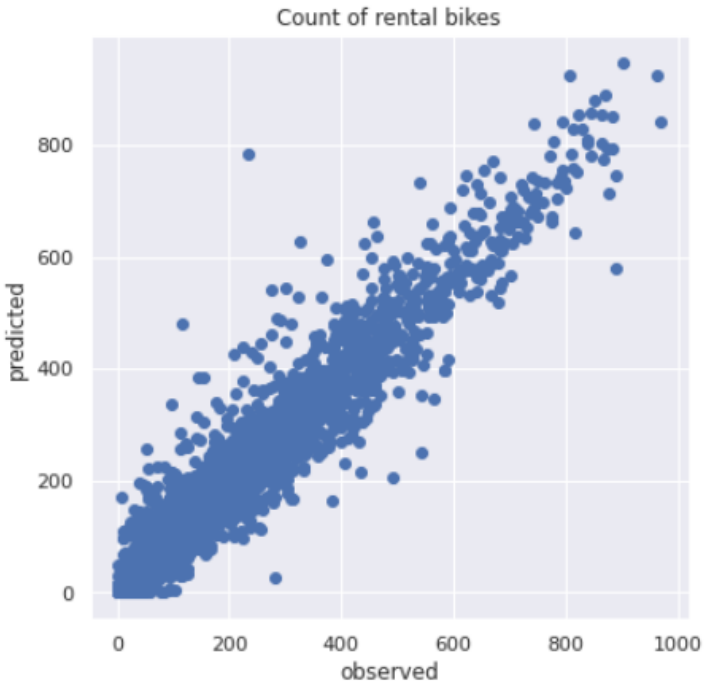
```
Best params: {'C': 1000, 'kernel': 'rbf'}
Best cross validaton score 0.9244912445605717
```

```
SVR(C=1000)
```

```
R2-score for train : 0.9666252878187059
RMSE for train : 33.2043882055002
```



```
R2-score for test : 0.9308838636967671
RMSE for test : 47.2664751995594
```



Текущее значение:

	R2_train	RMSE_train	R2_test	RMSE_test	R2_train (after tuning)	RMSE_train (after tuning)	R2_test (after tuning)	RMSE_test (after tuning)
Linear regression	0.702	99.153	0.689	100.245				
Ridge regression	0.702	99.155	0.689	100.243	0.702	99.155	0.689	100.244
Lasso	0.693	100.702	0.681	101.578	0.702	99.162	0.689	100.248
ElasticNet	0.603	114.449	0.598	114.050	0.702	99.167	0.689	100.253
SVM	0.465	132.908	0.474	130.395	0.967	33.204	0.931	47.266

Лучшие значения:

```
best_results
```

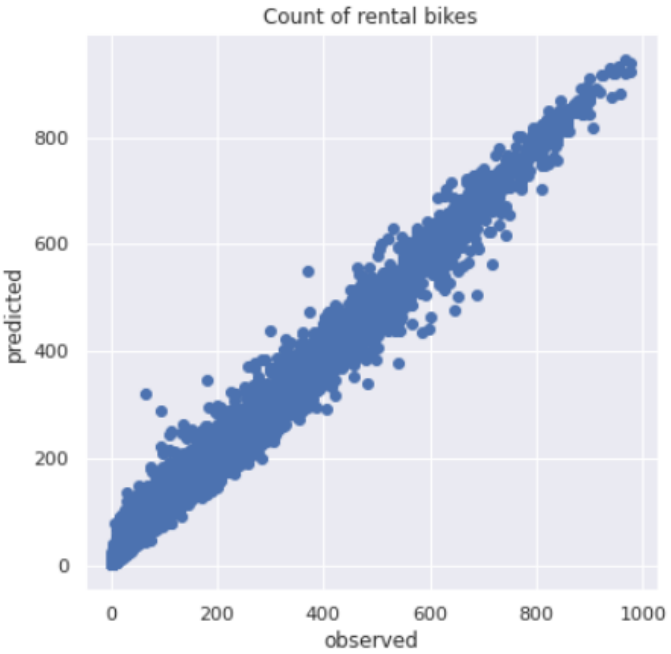
	R2_train	RMSE_train	R2_test	RMSE_test	Best parameters
Linear regression	0.702	99.153	0.689	100.245	-
Ridge regression	0.702	99.155	0.689	100.243	alpha_l = 1.024
Lasso	0.702	99.162	0.689	100.248	alpha_ll = 0.006
ElasticNet	0.702	99.167	0.689	100.253	alpha = 0.01, l1_ratio = 0.99
SVM	0.967	33.204	0.931	47.266	C = 1000, kernel = rbf

6) Random Forest

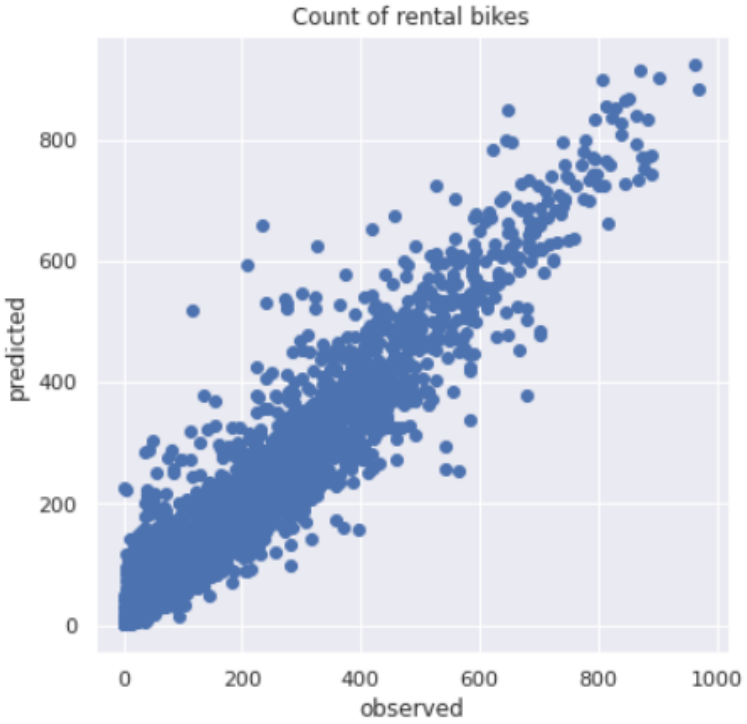
Метод, заключающийся в использовании комитета (ансамбля) решающих деревьев. Алгоритм сочетает в себе две основные идеи: метод бэггинга Бреймана, и метод случайных подпространств, предложенный Тин Кам Хо.

Без тьюнинга параметров:

```
R2-score for train : 0.9871128817062834
RMSE for train : 20.633106777777602
```




```
R2-score for test : 0.9103177057658728
RMSE for test : 53.84146217815935
```



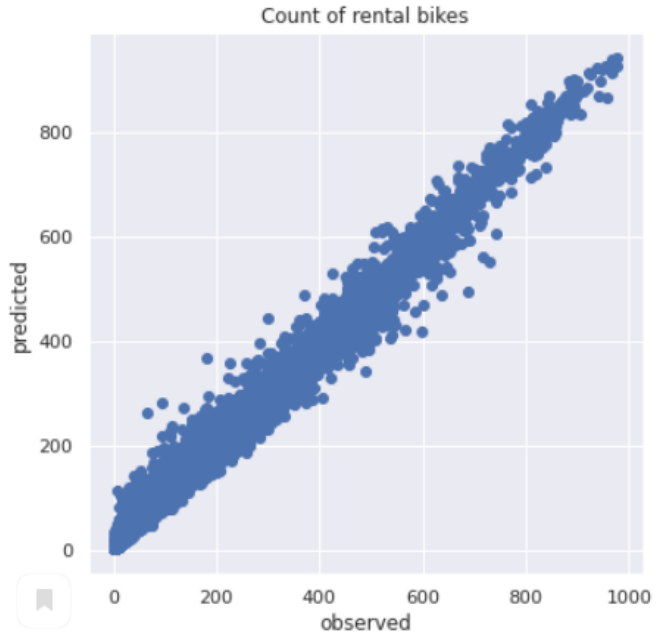
Настройка параметров
Параметры:

количество деревьев в лесу
количество предикторов, случайно выбранных при каждом разделении

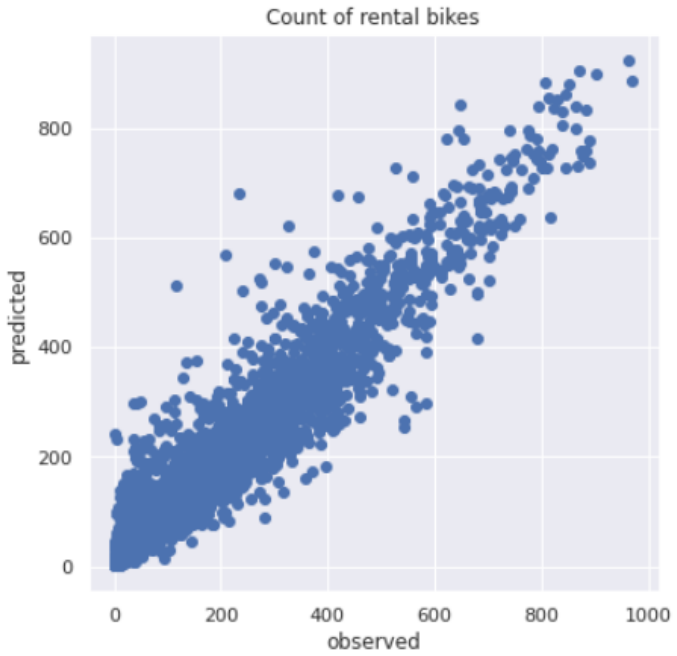
```
Best params: {'max_features': 33, 'n_estimators': 100}
Best cross validaton score 0.9004165738676448
```

```
RandomForestRegressor(max_features=33)
```

```
R2-score for train : 0.9869940164307962
RMSE for train : 20.728043848115583
```




```
R2-score for test : 0.9093800570615598
RMSE for test : 54.12219265356365
```

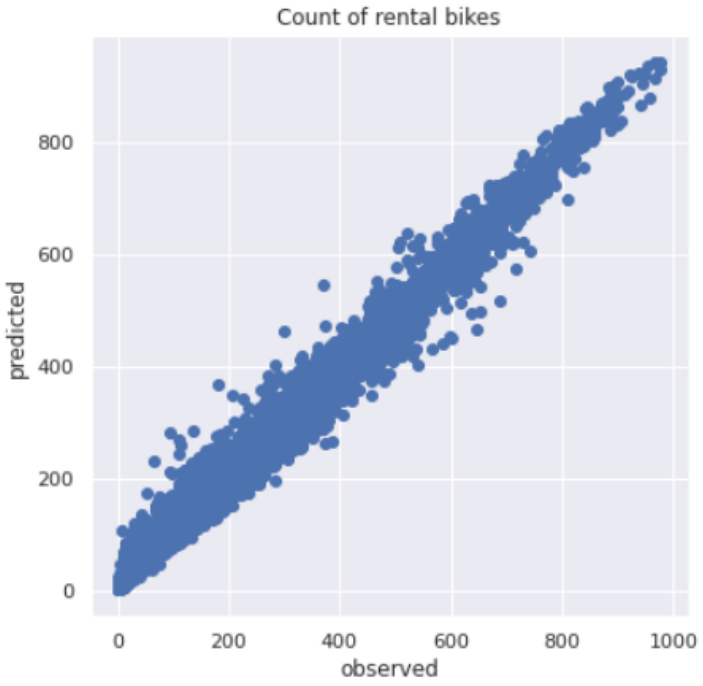


Вы можете видеть, что метрики после использования поиска по сетке стали немного хуже. Я пробовала запускать поиск по сетке с немного другими параметрами и в одном из случаев получила следующие лучшие параметры:

- n_estimators = 95
- max_features = 42

Используя эти параметры:

```
R2-score for train : 0.9870279082645154
RMSE for train : 20.701018991772013
```

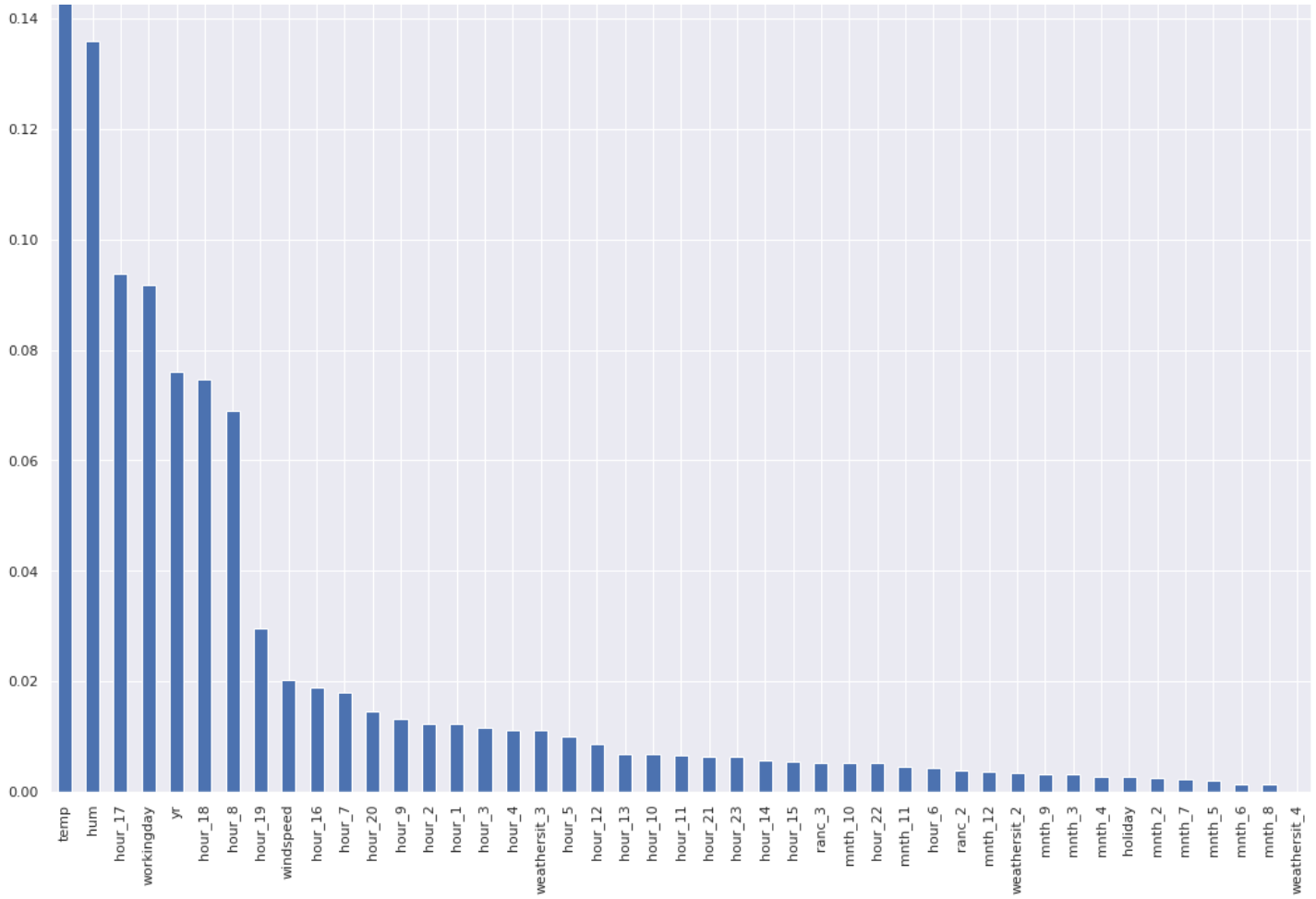



```
R2-score for test : 0.9108437886957557
RMSE for test : 53.683310917977344
```

Мы видим, что такое сочетание параметров дает наилучший результат на тестовом образце.

Теперь давайте взглянем на важность параметров.

	Importance
temp	0.164981
hum	0.135860
hour_17	0.093844
workingday	0.091822
yr	0.076052



Мы видим, что наиболее важными являются температура и влажность, но эти значения недостаточно велики, чтобы мы могли использовать только их для предсказания.

Текущие результаты:

	R2_train	RMSE_train	R2_test	RMSE_test	R2_train (after tuning)	RMSE_train (after tuning)	R2_test (after tuning)	RMSE_test (after tuning)
Linear regression	0.702	99.153	0.689	100.245				
Ridge regression	0.702	99.155	0.689	100.243	0.702	99.155	0.689	100.244
Lasso	0.693	100.702	0.681	101.578	0.702	99.162	0.689	100.248
ElasticNet	0.603	114.449	0.598	114.050	0.702	99.167	0.689	100.253
SVM	0.465	132.908	0.474	130.395	0.967	33.204	0.931	47.266
Random Forest	0.987	20.633	0.910	53.841	0.987	20.701	0.911	53.683

```
best_results
```

	R2_train	RMSE_train	R2_test	RMSE_test	Best parameters
Linear regression	0.702	99.153	0.689	100.245	-
Ridge regression	0.702	99.155	0.689	100.243	alpha_I = 1.024
Lasso	0.702	99.162	0.689	100.248	alpha_II = 0.006
ElasticNet	0.702	99.167	0.689	100.253	alpha = 0.01, l1_ratio = 0.99
SVM	0.967	33.204	0.931	47.266	C = 1000, kernel = rbf
Random Forest	0.987	20.701	0.911	53.683	n_estimators = 95, max_features = 42

7) Gradient Boosting

Будет использоваться XGBoost (Extreme Gradient Boosting), расширенная реализация алгоритма повышения градиента. XGBoost имеет три типа параметров, было решено выписать описание параметров для дальнейшего удобства создания и настройки модели:

Общие параметры:

booster: решено использовать древовидные модели, поэтому этот параметр не будет настраиваться.

тихий: тихий режим означает, что текущие сообщения не будут напечатаны

nthread: параметр используется для параллельной обработки;

Параметры бустера:

эта [по умолчанию = 0,3]: аналогично скорости обучения в GBM;

min_child_weight [по умолчанию = 6] : определяет минимальную сумму весов всех наблюдений, необходимых в дочернем элементе. Используется для контроля переобучения. Более высокие значения не позволяют модели изучать отношения, которые могут быть очень специфичными для конкретной выборки, выбранной для дерева.

max_depth [по умолчанию=6]: максимальная глубина дерева. Используется для управления переобучением, поскольку более высокая глубина позволит модели изучить отношения, очень специфичные для конкретного образца.

max_leaf_nodes: максимальное количество конечных узлов или листьев в дереве. Если это определено, GBM будет игнорировать max_depth, так как создаются бинарные деревья, глубина n создаст максимум 2n листьев.

гамма [по умолчанию = 0]: узел разделяется только тогда, когда результирующее разделение дает положительное снижение функции потерь. Гамма указывает минимальное снижение потерь, необходимое для разделения. Делает алгоритм консервативным. Значения могут варьироваться в зависимости от функции потерь и должны быть настроены.

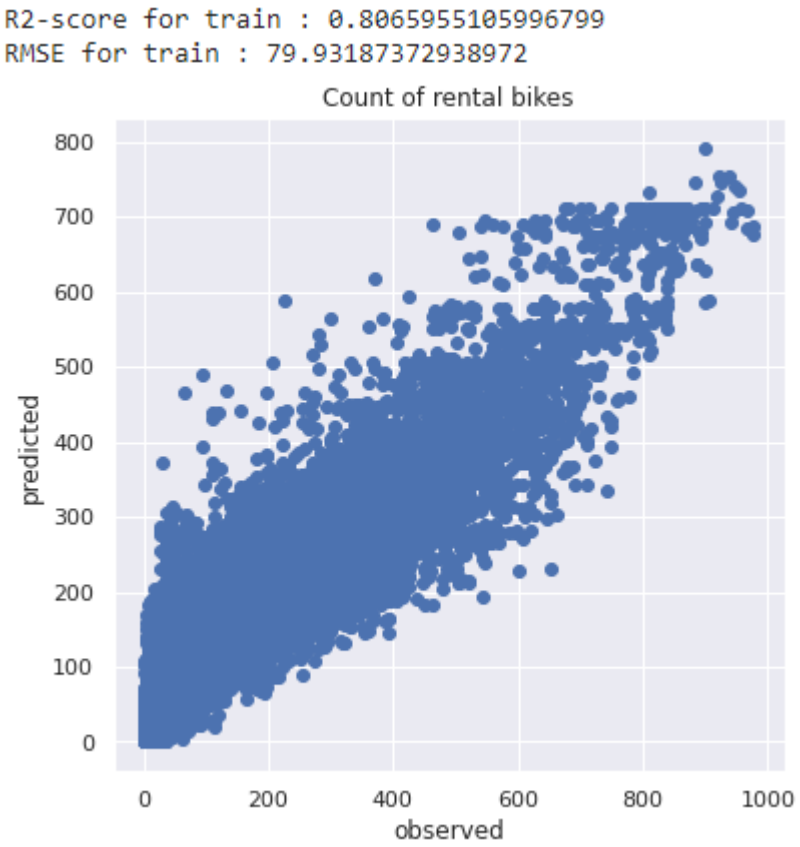
max_delta_step [по умолчанию = 0]: этот параметр обычно не используется в моделях на основе деревьев, но может помочь в логистической регрессии.

подвыборка [по умолчанию = 1]: то же, что и подвыборка GBM. Обозначает долю наблюдений, являющихся случайными выборками для каждого дерева. Более низкие значения делают алгоритм более консервативным и предотвращают переоснащение, но слишком малые значения могут привести к недостаточному подбору. Типичные значения: 0,5-1

colsample_bytree [по умолчанию = 1]: аналогично max_features в GBM. Обозначает долю столбцов для случайных выборок для каждого дерева. Типичные значения: 0,5-1

Параметры задачи обучения

До подбора параметров:

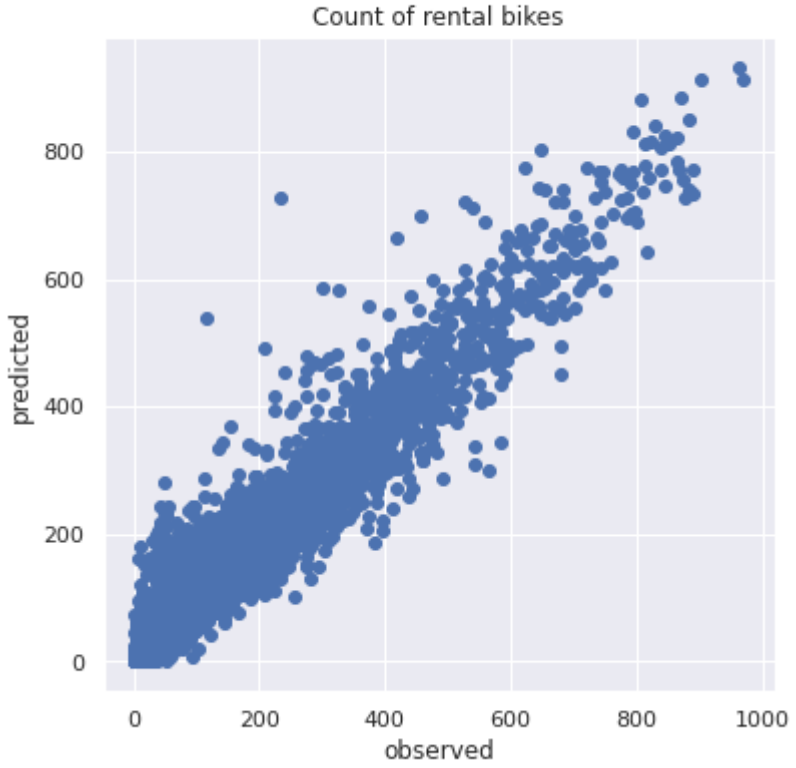


- Tuning. Three parameters were selected for tuning:

1. min_child_weight
2. max_depth
3. gamma

```
R2-score for train : 0.9467694045296203
RMSE for train : 41.93410520464002
```


```
R2-score for test : 0.9165096422201483
RMSE for test : 51.94953346872281
```

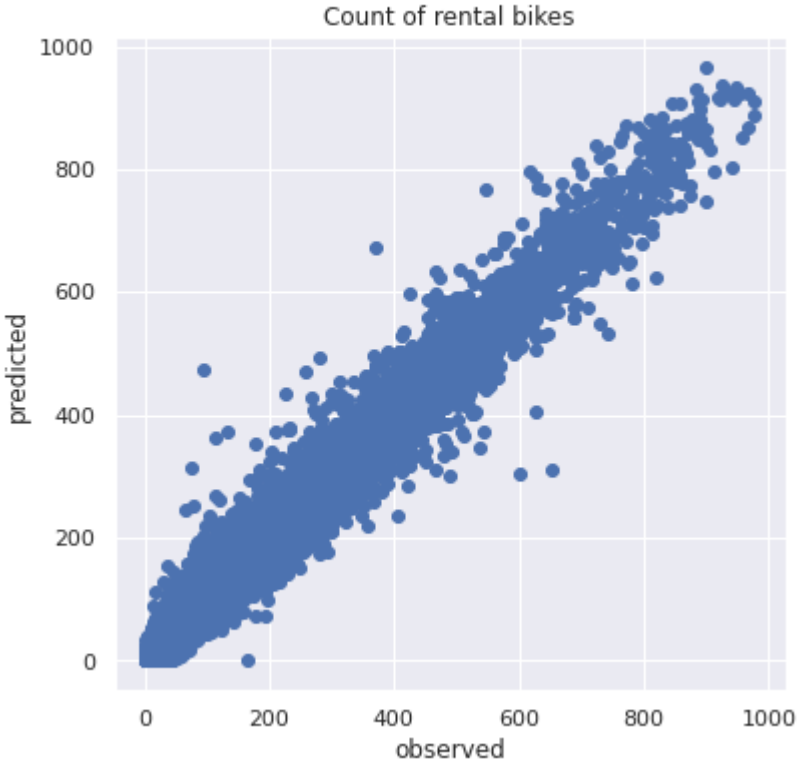


8) Neural Network

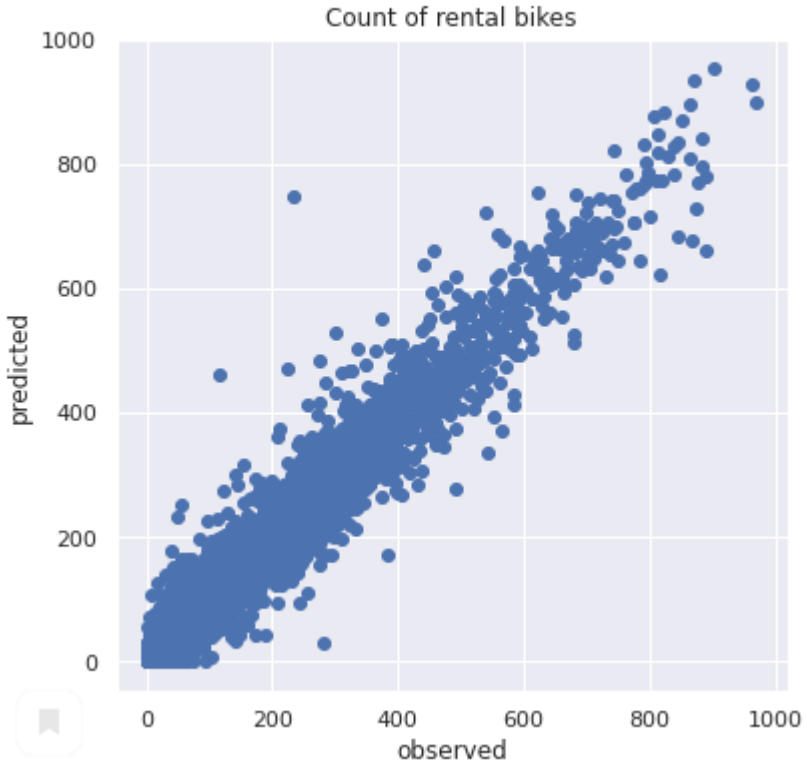
- Tuning parameters:
    1. Learning rate
    2. Number of epochs

```
Best params: {'epochs': 80, 'learning_rate': 0.05}
Best cross validaton score -2272.579214136912
```

```
R2-score for train : 0.9713076408216411
RMSE for train : 30.78717551547981
```



```
R2-score for test : 0.941379979103614
RMSE for test : 43.52978029064584
```




```
best_results
```

	R2_train	RMSE_train	R2_test	RMSE_test	Best parameters
Linear regression	0.702	99.153	0.689	100.245	-
Ridge regression	0.702	99.155	0.689	100.243	alpha_l = 1.024
Lasso	0.702	99.162	0.689	100.248	alpha_ll = 0.006
ElasticNet	0.702	99.167	0.689	100.253	alpha = 0.01, l1_ratio = 0.99
SVM	0.967	33.204	0.931	47.266	C = 1000, kernel = rbf
Random Forest	0.987	20.701	0.911	53.683	n_estimators = 95, max_features = 42
Gradient Boosting	0.947	41.934	0.917	51.950	max_depth = 8, min_child_weight = 4, gamma = 0.4
Neural Network	0.971	30.787	0.941	43.530	epochs = 80, learning_rate = 0.05

Выводы:

1) Нейросетевая модель показала лучшие результаты, чем модели других алгоритмов. Он имеет наибольший коэффициент детерминации R2 на тестовой выборке и наименьшую среднеквадратичную ошибку RMSE. Лучшие параметры настройки: Learning_rate=0.05 и epochs=80. Показатели NN в обучающей выборке лучше, чем в тесте, но незначительно, и это нормально, что производительность модели на обучающих данных выше.

2) Модель SVR также показала себя очень хорошо и показала второй результат по метрикам на тестовой выборке. Наилучшие параметры: значение стоимости C=1000 и ядро=rbf.

3)Модель Gradient Boosting занимает 3-е место с гиперпараметрами: максимальная глубина = 8, минимальная дочерняя _ масса = 4 и гамма = 0,4.

4) Хотя модель Random Forest имеет лучшие показатели на обучающих данных (R2=0,987!), но на тестовых данных она занимает лишь 4-е место. Можно предположить, что в данном случае имело место небольшое переоснащение модели. Однако модель по-прежнему показывает высокий результат на тестовых данных, поэтому я не могу быть уверен, что имеет место переобучение.

5) Температура и влажность являются наиболее важными характеристиками по результатам моделей Gradient Boosting и Random Forest.

6) Линейные модели (Линейная регрессия, Лассо, Ридж и Эластичная сетка) имеют наименьший R2-показатель (хуже более чем на 0,2, чем у моделей, перечисленных в предыдущих пунктах) и показатель RMSE для них примерно в 2 раза больше. Возможно, это связано с тем, что модели слишком просты для описания закономерностей в данных. Кроме того, в данных очень много категориальных признаков, тогда как реальные признаки практически отсутствуют. Все эти модели дают практически одинаковый результат и каких-либо существенных улучшений найти не удается. Различные виды регуляризации призваны помочь в борьбе с переоснащением и мультиколлинеарностью, в нашем случае оно было скорее недообучением. Попарно коррелированные признаки были обнаружены на этапе предварительной обработки, а один признак в паре был отброшен перед настройкой моделей.