Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИУ_____

КАФЕДРА ИУ5_____

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОМУ ПРОЕКТУ

НА ТЕМУ:

Решение задачи машинного обучения

Студент группы ИУ5-61Б _____ _____ Аушева Л.И.
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель курсового проекта _____ _____Гапанюк Ю.Е.
(Подпись, дата) (И.О.Фамилия)

Консультант _____ _____
(Подпись, дата) (И.О.Фамилия)

2020 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ
Заведующий кафедрой _____
(Индекс)
_____ _____
(И.О.Фамилия)
« _____ » _____ 20 ____ г.

З А Д А Н И Е
на выполнение курсового проекта

по дисциплине «Технологии машинного обучения»_____

Студент группы ИУ5-61Б_____

___Аушева Лиза Иссаевна_____
(Фамилия, имя, отчество)

Тема курсового проекта _____
_____
_____

Направленность КП (учебный, исследовательский, практический, производственный, др.)
_____
Источник тематики (кафедра, предприятие, НИР) _____

График выполнения проекта: 25% к ___ нед., 50% к ___ нед., 75% к __ нед., 100% к ___ нед.

Задание решение задачи машинного обучения на основе материалов дисциплины. Выполняется студентом единолично.
_____
_____
_____
_____

Оформление курсового проекта:

Расчетно-пояснительная записка на __18___ листах формата А4.
Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)
_____
_____
_____
_____

Дата выдачи задания « 12 » февраля 2020 г.

Руководитель курсового проекта _____ _____Аушева Л.И.
(Подпись, дата) (И.О.Фамилия)

Студент _____ _____Гапанюк Ю.Е.
(Подпись, дата) (И.О.Фамилия)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

Оглавление

Задание установленного образца

Схема типового исследования, проводимого студентом в рамках курсовой работы, содержит выполнение следующих шагов:

- Поиск и выбор набора данных для построения моделей машинного обучения. На основе выбранного набора данных студент должен построить модели машинного обучения для решения или задачи классификации, или задачи регрессии.
- Проведение разведочного анализа данных. Построение графиков, необходимых для понимания структуры данных. Анализ и заполнение пропусков в данных.
- Выбор признаков, подходящих для построения моделей. Кодирование категориальных признаков Масштабирование данных. Формирование вспомогательных признаков, улучшающих качество моделей.
- Проведение корреляционного анализа данных. Формирование промежуточных выводов о возможности построения моделей машинного обучения. В зависимости от набора данных, порядок выполнения пунктов 2, 3, 4 может быть изменен.
- Выбор метрик для последующей оценки качества моделей. Необходимо выбрать не менее трех метрик и обосновать выбор.
- Выбор наиболее подходящих моделей для решения задачи классификации или регрессии. Необходимо использовать не менее пяти моделей, две из которых должны быть ансамблевыми.
- Формирование обучающей и тестовой выборок на основе исходного набора данных.
- Построение базового решения (baseline) для выбранных моделей без подбора гиперпараметров. Производятся

обучение моделей на основе обучающей выборки и оценка качества моделей на основе тестовой выборки.

- Подбор гиперпараметров для выбранных моделей. Рекомендуется использовать методы кросс-валидации. В зависимости от используемой библиотеки можно применять функцию GridSearchCV, использовать

  перебор параметров в цикле, или использовать другие методы.

- Повторение пункта 8 для найденных оптимальных значений гиперпараметров. Сравнение качества полученных моделей с качеством baseline-моделей.

- Формирование выводов о качестве построенных моделей на основе выбранных метрик. Результаты сравнения качества рекомендуется отобразить в виде графиков и сделать выводы в форме текстового описания. Рекомендуется построение графиков обучения и валидации, влияния значений гиперпарметров на качество моделей и т.д.

Введение

Курсовой проект – самостоятельная часть учебной дисциплины «Технологии машинного обучения» – учебная и практическая исследовательская студенческая работа, направленная на решение комплексной задачи машинного обучения. Результатом курсового проекта является отчет, содержащий описания моделей, тексты программ и результаты экспериментов.

Курсовой проект опирается на знания, умения и владения, полученные студентом в рамках лекций и лабораторных работ по дисциплине.

Основная часть. Описание постановки задачи и последовательности действий по решению поставленной задачи

Описание набора данных

В данной работе для исследований был выбран следующий датасет:

https://www.kaggle.com/c/bike-sharing-demand

Задача: ***По историческим данным о прокате велосипедов и погодным условиям необходимо оценить спрос на прокат велосипедов.***

Файл:

Входные переменные (на основе физико-химических тестов):

***datetime** - hourly date + timestamp*

***season** - 1 = spring, 2 = summer, 3 = fall, 4 = winter*

***holiday** - whether the day is considered a holiday*

***workingday** - whether the day is neither a weekend nor holiday*

***weather** - 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** - temperature in Celsius*

***atemp** - "feels like" temperature in Celsius*

***humidity** - relative humidity*

***windspeed** - wind speed*

***casual** - number of non-registered user rentals initiated*

***registered** - number of registered user rentals initiated*

***count** - number of total rentals*

Выходная переменная (на основе отзывов потребителей):

Спрос

Для данного набора данных мы будем решать задачу регрессии.

Ход работы

Импортируем необходимые для работы библиотеки:

In [2]:
```
import warnings
warnings.filterwarnings('ignore')
```

Sklearn

Bike sharing demand

Задача на kaggle: https://www.kaggle.com/c/bike-sharing-demand

По историческим данным о прокате велосипедов и погодным условиям необходимо оценить спрос на прокат велосипедов.

В исходной постановке задачи доступно 11 признаков: https://www.kaggle.com/c/prudential-life-insurance-assessment/data

В наборе признаков присутсвуют вещественные, категориальные, и бинарные данные.

Для демонстрации используется обучающая выборка из исходных данных train.csv, файлы для работы прилагаются.

Библиотеки

In [4]:
```
from sklearn import model_selection, linear_model, metrics

import numpy as np
import pandas as pd
```

In [5]:
```
%pylab inline
```

```
Populating the interactive namespace from numpy and matplotlib
```

Загрузка данных

In [8]:
```
raw_data = pd.read_csv('datasets/bike_sharing_demand.csv',sep=',')
```

In [9]:
```
raw_data.head()
```

Out[9]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	3	13	16
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	8	32	40
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	5	27	32
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0	3	10	13
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0	0	1	1

datetime - hourly date + timestamp

season - 1 = spring, 2 = summer, 3 = fall, 4 = winter

holiday - whether the day is considered a holiday

workingday - whether the day is neither a weekend nor holiday

weather - 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 - temperature in Celsius

atemp - "feels like" temperature in Celsius

humidity - relative humidity

windspeed - wind speed

casual - number of non-registered user rentals initiated

registered - number of registered user rentals initiated

count - number of total rentals

In [10]:
```
raw_data.shape
```

Out[10]:
```
(10886, 12)
```

In [12]:
```
raw_data.isnull().values.any()
```

Out[12]:
```
False
```

Предобработка данных

Типы признаков

In [13]:
```
raw_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10886 entries, 0 to 10885
Data columns (total 12 columns):
datetime      10886 non-null object
season        10886 non-null int64
holiday       10886 non-null int64
workingday    10886 non-null int64
weather       10886 non-null int64
temp          10886 non-null float64
atemp         10886 non-null float64
humidity      10886 non-null int64
windspeed     10886 non-null float64
casual        10886 non-null int64
registered    10886 non-null int64
count         10886 non-null int64
dtypes: float64(3), int64(8), object(1)
memory usage: 1020.7+ KB
```

Поменяем тип у *datetime*. С object-> datetime

In [17]:
```
raw_data.datetime = raw_data.datetime.apply(pd.to_datetime)
raw_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10886 entries, 0 to 10885
Data columns (total 12 columns):
datetime      10886 non-null datetime64[ns]
season        10886 non-null int64
holiday       10886 non-null int64
workingday    10886 non-null int64
weather       10886 non-null int64
temp          10886 non-null float64
atemp         10886 non-null float64
humidity      10886 non-null int64
windspeed     10886 non-null float64
casual        10886 non-null int64
registered    10886 non-null int64
count         10886 non-null int64
dtypes: datetime64[ns](1), float64(3), int64(8)
memory usage: 1020.7 KB
```

Создадим два новых признака: *месяц и час, когда это происходит*

In [25]:
```
raw_data['month'] = raw_data.datetime.apply(lambda x: x.month)
raw_data['hour'] = raw_data.datetime.apply(lambda x: x.hour)
```

In [26]:
```
raw_data.head()
```

Out[26]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count	month	hour
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	3	13	16	1	0
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	8	32	40	1	1
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	5	27	32	1	2
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0	3	10	13	1	3
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0	0	1	1	1	4

Обучение и отложенный тест

Обучающая выборка для создания модели и обучения ее. Тестовая выборка - для проверки качества модели

In [27]:
```
train_data = raw_data.iloc[:-1000, :]
hold_out_test_data = raw_data.iloc[-1000:, :]
```

In [29]:
```
raw_data.shape, train_data.shape, hold_out_test_data.shape
```

Out[29]:
```
((10886, 14), (9886, 14), (1000, 14))
```

In [33]:
```
print('train period from {} to {}'.format(train_data.datetime.min(), train_data.datetime.max()))
print('evaluation period from {} to {}'.format(hold_out_test_data.datetime.min(), hold_out_test_data.datetime.max()))
```

```
train period from 2011-01-01 00:00:00 to 2012-10-16 06:00:00
evaluation period from 2012-10-16 07:00:00 to 2012-12-19 23:00:00
```

Данные и целевая функция

In [38]:
```
## обучение
train_labels = train_data['count'].values
train_data = train_data.drop(['datetime', 'count'], axis=1)
train_data.head()
```

Out[38]:

	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	month	hour
0	1	0	0	1	9.84	14.395	81	0.0	3	13	1	0
1	1	0	0	1	9.02	13.635	80	0.0	8	32	1	1
2	1	0	0	1	9.02	13.635	80	0.0	5	27	1	2
3	1	0	0	1	9.84	14.395	75	0.0	3	10	1	3
4	1	0	0	1	9.84	14.395	75	0.0	0	1	1	4

In [40]:
```
## test
test_labels = hold_out_test_data['count'].values
test_data = hold_out_test_data.drop(['count', 'datetime'], axis=1)
test_data.head()
```

Out[40]:

	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	month	hour
9886	4	0	1	1	17.22	21.210	67	6.0032	20	505	10	7
9887	4	0	1	1	18.04	21.970	62	0.0000	35	800	10	8
9888	4	0	1	1	19.68	23.485	55	16.9979	32	323	10	9
9889	4	0	1	1	20.50	24.240	48	19.0012	65	157	10	10
9890	4	0	1	1	20.50	24.240	45	27.9993	56	172	10	11

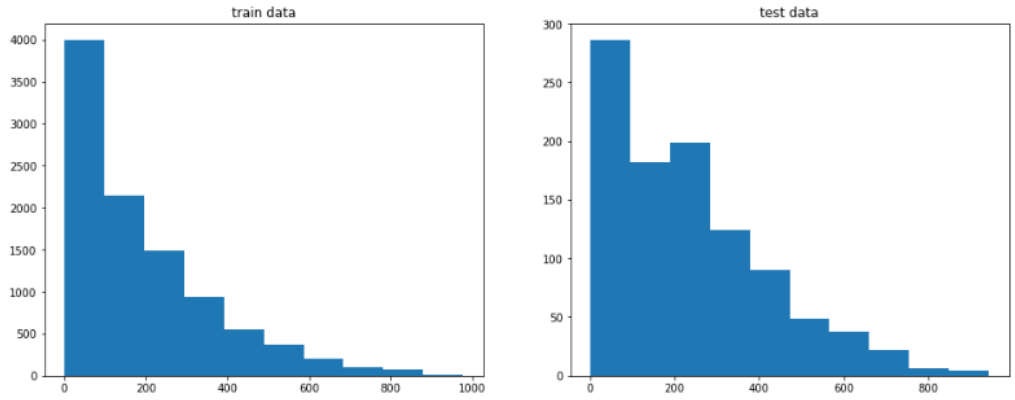
Целевая функция на обучающей выборке и на отложенном тесте

In [41]:
```
pylab.figure(figsize=(16,6))

pylab.subplot(1, 2, 1)
pylab.hist(train_labels)
pylab.title('train data')

pylab.subplot(1, 2, 2)
pylab.hist(test_labels)
pylab.title('test data')
```

Out[41]: Text(0.5, 1.0, 'test data')



Числовые признаки

In [43]:
```
numeric_columns = ['temp', 'atemp', 'humidity', 'windspeed', 'casual', 'registered', 'month', 'hour']
```

In [45]:
```
train_data = train_data[numeric_columns]
test_data = test_data[numeric_columns]
```

Числовые признаки

In [43]:
```
numeric_columns = ['temp', 'atemp', 'humidity', 'windspeed', 'casual', 'registered', 'month', 'hour']
```

In [45]:
```
train_data = train_data[numeric_columns]
test_data = test_data[numeric_columns]
```

In [46]:
```
train_data.head()
```

Out[46]:

	temp	atemp	humidity	windspeed	casual	registered	month	hour
0	9.84	14.395	81	0.0	3	13	1	0
1	9.02	13.635	80	0.0	8	32	1	1
2	9.02	13.635	80	0.0	5	27	1	2
3	9.84	14.395	75	0.0	3	10	1	3
4	9.84	14.395	75	0.0	0	1	1	4

In [47]:
```
test_data.head()
```

Out[47]:

	temp	atemp	humidity	windspeed	casual	registered	month	hour
9886	17.22	21.210	67	6.0032	20	505	10	7
9887	18.04	21.970	62	0.0000	35	800	10	8
9888	19.68	23.485	55	16.9979	32	323	10	9
9889	20.50	24.240	48	19.0012	65	157	10	10
9890	20.50	24.240	45	27.9993	56	172	10	11

Модель

Так как у нас регрессия, то обучим регрессор. Моделью будет регрессор на основе стохастического градиентного спуска

In [48]:
```
regressor = linear_model.SGDRegressor(random_state=0)
```

In [50]:
```
regressor.fit(train_data, train_labels)
metrics.mean_absolute_error(test_labels, regressor.predict(test_data))
```

Out[50]:
```
32678333066101.656
```

Невероятно большая ошибка.

Посмотрим на истинные метки, и наши

In [51]:
```
test_labels[:10]
```

Out[51]:
```
array([525, 835, 355, 222, 228, 325, 328, 308, 346, 446], dtype=int64)
```

In [52]:
```
regressor.predict(test_data)[:10]
```

Out[52]:
```
array([-8.15217468e+13, -1.27940348e+14, -5.15430820e+13, -2.29149119e+13,
       -2.58255957e+13, -3.85635403e+13, -3.92648763e+13, -3.03104198e+13,
       -4.03846575e+13, -5.57515848e+13])
```

У нас очень плохие значения.

Посмотрим на наши коэффициенты.

In [53]:
```
regressor.coef_
```

Out[53]:
```
array([-2.30285477e+10, -5.87204402e+08, -1.27291755e+10, -1.27359409e+09,
        5.83730183e+10, -1.60478942e+11, -3.67459548e+10, -4.87006373e+09])
```

Коэффициенты просто нереальны. Это происходит из-за того, что мы не отмасштабировали признаки

Scaling

In [55]:
```
from sklearn.preprocessing import StandardScaler
```

Создаем $scaler$

Чтобы применить наше преобразование, нужно сначала его обучить(то есть высчитать параметры μ и σ)

Обучать $scaler$ можно только на обучающей выборке(потому что часто на практике нам неизвестна тестовая выборка)

In [60]:
```
## создаем scaler
scaler = StandardScaler()
scaler.fit(train_data, train_labels)
scaled_trained_data = scaler.transform(train_data)
scaled_test_data = scaler.transform(test_data)
```

Теперь можно снова обучить модель

In [61]:
```
regressor.fit(scaled_trained_data, train_labels)
metrics.mean_absolute_error(test_labels, regressor.predict(scaled_test_data))
```

Out[61]:
```
0.042930483012408885
```

Ошибка получилось очень маленька

Посмотрим на целефую функцию и наши прогнозы

In [62]:
```
test_labels[:10]
```

Out[62]:
```
array([525, 835, 355, 222, 228, 325, 328, 308, 346, 446], dtype=int64)
```

In [64]:
```
regressor.predict(scaled_test_data)[:10]
```

Out[64]:
```
array([524.90958201, 834.88816062, 354.94091402, 221.96933203,
       227.95283001, 324.96044934, 327.966113  , 307.98463956,
       345.96942383, 445.96231877])
```

Мы ошибаемся меньше чем на 1 велосипед, это очень странно.

Посмотрим на коэффициенты

In [65]:
```
regressor.coef_
```

Out[65]:
```
array([ 4.58902678e-01, -4.51836158e-01,  6.62608792e-04, -1.40703258e-02,
        5.08590377e+01,  1.48008168e+02, -1.32281341e-03,  7.59230341e-03])
```

In [66]:
```
print(list(map(lambda x: round(x, 2), regressor.coef_)))
```

```
[0.46, -0.45, 0.0, -0.01, 50.86, 148.01, -0.0, 0.01]
```

Видно, что почти все признаки принимают маленькие коэффициенты, за исключением двух.

Посмотрим на эти признаки

In [67]:
```
train_data.head()
```

Out[67]:

	temp	atemp	humidity	windspeed	casual	registered	month	hour
0	9.84	14.395	81	0.0	3	13	1	0
1	9.02	13.635	80	0.0	8	32	1	1
2	9.02	13.635	80	0.0	5	27	1	2
3	9.84	14.395	75	0.0	3	10	1	3
4	9.84	14.395	75	0.0	0	1	1	4

Это признаки ***casual и registered***

In [69]:
```
train_labels[:10]
```

Out[69]:
```
array([16, 40, 32, 13,  1,  1,  2,  3,  8, 14], dtype=int64)
```

Видно, что если сложить два эти признака, то получим целевую функцию

In [71]:
```
np.all(train_data.casual + train_data.registered == train_labels)
```

Out[71]:
```
True
```

То есть мы использовали те данные, по которым однозначно восстанавливается целевая функция

Удалим из нашей выборки эти признаки

In [72]:
```
train_data.drop(['casual', 'registered'], axis=1, inplace=True)
test_data.drop(['casual', 'registered'], axis=1, inplace=True)
```

Отмасштабируем признаки на новом наборе данных.

И обучим модель

In [74]:
```
scaler.fit(train_data, train_labels)
scaled_train_data = scaler.transform(train_data)
scaled_test_data = scaler.transform(test_data)
```

In [77]:
```
regressor.fit(scaled_train_data, train_labels)
metrics.mean_absolute_error(test_labels, regressor.predict(scaled_test_data))
```

Out[77]:
```
121.8835371361759
```

In [78]:
```
print(list(map(lambda x: round(x, 2), regressor.coef_)))
```

```
[30.01, 32.15, -42.28, 3.78, 12.71, 50.06]
```

Теперь видно, что почти все признаки вносят вклад в модель. Веса похожи на правильные

Мы построили модель. Теперь попытаемся ее улучшить

Для этого проведем кросс-валидацию. Но тут существует проблема: нам необходимо масштабировать данные. То есть нужно для каждого фолда провести масштабирование, а потом обучить его и проверить качество. Вся реализация будет очень громоздкой.

Благо, существует класс $Pipeline$

Pipilene

In [79]:
```
from sklearn.pipeline import Pipeline
```

Вместо одного преобразования, $Pipeline$ позволяет делать целую цепочку преобразований

- каждый шаг представляется $tuple$
- первый элемент: название шага, второй элемент: объект, который способен преобразовывать данные
- ***главное, чтобы у объектов были такие методы как fit и $transform$***

In [82]:
```
## создаем Pipeline из двух шагов: scaling и классификация
pipeline = Pipeline(steps=[('scaling', scaler), ('regression', regressor)])
```

In [83]:
```
pipeline.fit(train_data, train_labels)
metrics.mean_absolute_error(test_labels, pipeline.predict(test_data))
```

Out[83]:
```
121.8835371361759
```

Подбор параметров

Параметры будем подбирать по сетке

Посмотрим сначала как правильно к ним обращаться

In [88]:
```
pipeline.get_params().keys()
```

Out[88]:
```
dict_keys(['memory', 'steps', 'verbose', 'scaling', 'regression', 'scaling__copy', 'scaling__with_mean', 'scaling__with_std',
```

In [89]:
```
parameters_grid = {
    'regression__loss' : ['huber', 'epsilon_insensitive', 'squared_loss', ],
    'regression__max_iter' : [3, 5, 10, 50],
    'regression__penalty' : ['l1', 'l2', 'none'],
    'regression__alpha' : [0.0001, 0.01],
    'scaling__with_mean' : [0., 0.5],
}
```

Строим сетку

In [92]:
```
grid_cv = model_selection.GridSearchCV(pipeline, parameters_grid, scoring='neg_mean_absolute_error', cv=4)
```

Обучаем сетку

In [93]:
```
%%time
grid_cv.fit(train_data, train_labels)
```

```
Wall time: 9.36 s
```

Out[93]:
```
GridSearchCV(cv=4, error_score=nan,
             estimator=Pipeline(memory=None,
                                steps=[('scaling',
                                        StandardScaler(copy=True,
                                                       with_mean=True,
                                                       with_std=True)),
                                       ('regression',
                                        SGDRegressor(alpha=0.0001,
                                                     average=False,
                                                     early_stopping=False,
                                                     epsilon=0.1, eta0=0.01,
                                                     fit_intercept=True,
                                                     l1_ratio=0.15,
                                                     learning_rate='invscaling',
                                                     loss='squared_loss',
                                                     max_iter=1000,
                                                     n_iter_no_change=5,
                                                     p...
             iid='deprecated', n_jobs=None,
             param_grid={'regression__alpha': [0.0001, 0.01],
                         'regression__loss': ['huber', 'epsilon_insensitive',
                                              'squared_loss'],
                         'regression__max_iter': [3, 5, 10, 50],
```

In [95]:
```
print(grid_cv.best_score_)
print(grid_cv.best_params_)
```

```
-108.61772632999148
{'regression__alpha': 0.01, 'regression__loss': 'squared_loss', 'regression__max_iter': 3, 'regression__penalty': 'l2', 'scal
```

Оценка по отложенному тесту

In [96]:
```
metrics.mean_absolute_error(test_labels, grid_cv.best_estimator_.predict(test_data))
```

Out[96]:
```
119.98978845935378
```

Наша ошибка теперь 120. То есть мы ошибаемся на 120 велосипедов

Посмотрим насколько наша ошибка большая, относительно среднего значения целевой переменной

In [97]:
```
np.mean(test_labels)
```

Out[97]:
```
232.159
```

Это плохо

Раньше ошибка была 121, теперь 120. По сути, наша оптимизация нам никак не помогла, то есть мы никак не улучшили модель

Посмотрим на значения наших предсказаний

In [98]:
```
test_labels[:10]
```

Out[98]:
```
array([525, 835, 355, 222, 228, 325, 328, 308, 346, 446], dtype=int64)
```

In [100]:
```
grid_cv.best_estimator_.predict(test_data)[:10]
```

Out[100]:
```
array([139.60470681, 159.80765341, 207.55935972, 237.76288054,
       257.83836668, 267.44558034, 272.49537469, 297.70688522,
       304.29818873, 313.58821156])
```

Видно, что отличия довольно серьезные

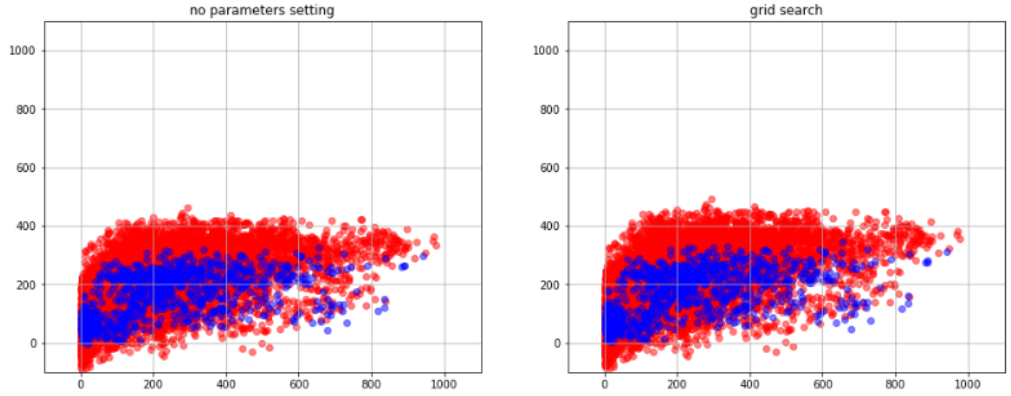
Построим график наших предсказаний на целевой метке. Идеально было бы, если наши прогнозы и целевая метка совпадали и образовывали диагональ

In [108]:
```
pylab.figure(figsize=(16,6))

pylab.subplot(1, 2, 1)
pylab.grid(True)
pylab.scatter(train_labels, pipeline.predict(train_data), alpha=0.5, color='red')
pylab.scatter(test_labels, pipeline.predict(test_data), alpha=0.5, color='blue')
pylab.title('no parameters setting')
pylab.xlim(-100, 1100)
pylab.ylim(-100, 1100)

pylab.subplot(1, 2, 2)
pylab.grid(True)
pylab.scatter(train_labels, grid_cv.best_estimator_.predict(train_data), alpha=0.5, color='red')
pylab.scatter(test_labels, grid_cv.best_estimator_.predict(test_data), alpha=0.5, color='blue')
pylab.title('grid search')
pylab.xlim(-100, 1100)
pylab.ylim(-100, 1100)
```

Out[108]:
```
(-100, 1100)
```



Другая модель

In [32]:
```
from sklearn.ensemble import RandomForestRegressor
```

In [33]:
```
regressor = RandomForestRegressor(random_state = 0, max_depth = 20, n_estimators = 50)
```

In [34]:
```
estimator = pipeline.Pipeline(steps = [
    ('feature_processing', pipeline.FeatureUnion(transformer_list = [
            #binary
            ('binary_variables_processing', preprocessing.FunctionTransformer(lambda data: data[:, binary_
data_indices])),

            #numeric
            ('numeric_variables_processing', pipeline.Pipeline(steps = [
                ('selecting', preprocessing.FunctionTransformer(lambda data: data[:, numeric_data_indice
s])),
                ('scaling', preprocessing.StandardScaler(with_mean = 0, with_std = 1))
                        ])),

            #categorical
            ('categorical_variables_processing', pipeline.Pipeline(steps = [
                ('selecting', preprocessing.FunctionTransformer(lambda data: data[:, categorical_data_indi
ces])),
                ('hot_encoding', preprocessing.OneHotEncoder(handle_unknown = 'ignore'))
                        ])),
        ])),
    ('model_fitting', regressor)
    ]
)
```

In [35]:
```
estimator.fit(train_data, train_labels)
```

Out[35]:
```
Pipeline(memory=None,
     steps=[('feature_processing', FeatureUnion(n_jobs=None,
       transformer_list=[('binary_variables_processing', FunctionTransformer(accept_sparse=False, check_in
verse=True,
          func=<function <lambda> at 0x7f7bf01f8f28>, inv_kw_args=None,
          inverse_func=None, kw_args=None, pass_y='dep...timators=50, n_jobs=None,
           oob_score=False, random_state=0, verbose=0, warm_start=False))])
```

In [36]:
```
metrics.mean_absolute_error(test_labels, estimator.predict(test_data))
```

Out[36]:
```
79.49758619912876
```

In [37]:
```
test_labels[:10]
```

Out[37]:
```
array([525, 835, 355, 222, 228, 325, 328, 308, 346, 446])
```

In [38]:
```
estimator.predict(test_data)[:10]
```

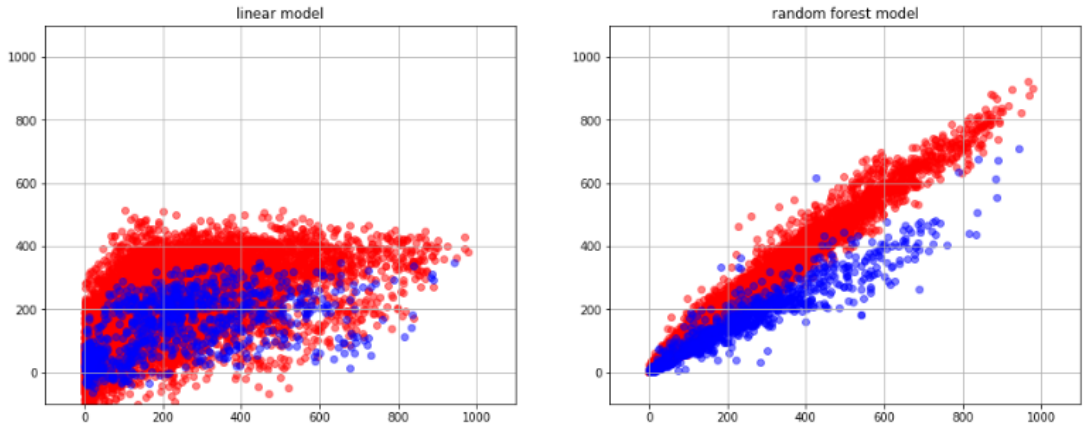
Out[38]:
```
array([409.565     , 505.56      , 256.06066667, 165.6502619 ,
       205.82474784, 265.91960439, 254.61615818, 317.85774486,
       280.98963263, 434.51333333])
```

In [39]:
```
pylab.figure(figsize=(16, 6))

pylab.subplot(1,2,1)
pylab.grid(True)
pylab.xlim(-100,1100)
pylab.ylim(-100,1100)
pylab.scatter(train_labels, grid_cv.best_estimator_.predict(train_data), alpha=0.5, color = 'red')
pylab.scatter(test_labels, grid_cv.best_estimator_.predict(test_data), alpha=0.5, color = 'blue')
pylab.title('linear model')

pylab.subplot(1,2,2)
pylab.grid(True)
pylab.xlim(-100,1100)
pylab.ylim(-100,1100)
pylab.scatter(train_labels, estimator.predict(train_data), alpha=0.5, color = 'red')
pylab.scatter(test_labels, estimator.predict(test_data), alpha=0.5, color = 'blue')
pylab.title('random forest model')
```

Out[39]:
```
Text(0.5, 1.0, 'random forest model')
```



Выводы по проделанной работе

В ходе курсовой работы были закреплены полученные в течение курса знания и навыки.

Список использованных источников

1. Конспект лекций по дисциплине “Технологии машинного обучения”. 2020: https://github.com/ugapanyuk/ml_course_2020/wiki/COURSE_TMO
2. Документация scikit-learn: https://scikit-learn.org/stable/index.html
3. Метрики в задачах машинного обучения: https://habr.com/ru/company/ods/blog/328372/