Предсказание температуры во времени и пространстве

Дробин М.Е. (МФТИ ГУ)

<2019-04-08 Пн>

Contents

Abstract

В этой работе исследуется качество прогнозирования температуры для одной из метерологической станций, обучая нейросетевые алгоритмы(RBF, MLP) и классические(KNN, spatial averaging, inverse distance methods) по данных от других станций в пределах Англии.

1 Введение

Температура воздуха - временные ряды с высоким временным разрешением - измеряют только в немногочисленных, далеко разнесенных друг от друга метерологических станциях. Поэтому появляется необходимость в пространственной интерполяции этих значений на области, где температура. Кроме пространственной интерполяции появляется необходимость в прогнозировании будущей темпертуры воздуха, т.е. временной интерполяции. Такие задачи появляются в сельском хозяйстве, прогнозировании погоды в городских условиях и т.д.

В качестве данных используется почасовая информация(влажность, ...) с 30 метерологических станций Англии(каждая в отдельном городе) за 2 года. И для некоторых из них предсказывается температура. Данные взяты из сервиса darksky.

2 Постановка задачи

Даны временные ряды $temp_L(t)$ - температура воздуха, замеренная на одной из погодных станций в Лондоне, и $\vec{x_j}(t), j \in \{1, 2, ...\}$ - климатические данные городов в окрестности Лондона (температура, давление, ...). Необходимо предсказать $temp_L(t+1)$

Обобщенная модель для решения этой задачи выглядит следующим образом: $temp_L(t+1) = f(temp_L(t), \vec{x_1}(t), \vec{x_2}(t), ...)$. Исследуется зависимость f от близости городов в окрестности Лондона к самому Лондону. Также исследуется качество известных моделей машинного обучениня примененых к этим данным. Приводится обоснование выбора функции потерь для этой задачи, а потому, выводы, построенные по качеству и некоторым характеристикам примененных моделей зависят не от моделей, а от самих данных.

3 Обзор литературы

В статье [1] авторы сравнили качество MLP и spatial average, knn, inverse distance methods для задачи интерполяции температуры на 11 NOAA станций, на которых измеряют температуру. Эти станции расположены примерно в одном штате. Температура с этих станций - это множество ответов алгоритма, а данные с GCM(general circulation model) - численное модели - это мн-возможным объектов. Такая задача называется downscaling GCM. Трудность заключается в том, что пространственное разрешение выходных данных GCM - от 2.5° × 2.5° до 8° × 10° в долготу и ширину - это слишком грубо для предсказания - поэтому обучают по такой сетке нейросеть и предсказывают температуру для точек между сетки - выхода GCM. Авторы предсказывали максимальную температуру за день T_{max} и посчитали качество нейросети(архите- ктура нейросети: 4-30-11 и 16-54-11 с сигмоидной функцией активацией). Для сетки с 4мя входами R^2 и rmse были в среднем по 11 станциям 5.69 и 0.93; для сетки с 16ю входами - 5.12 и 0.94.

В статье [2] авторы использовали почасовые данные с шести метерологических станций за 2 года с равнины, расположенной в Китае и ограниченной горами. 5 из них входят в мн-во объектов алгоритмов, для шестой предсказывается температу. 60% данных использовалось для обучения, остальное для теста. Сравнили MLP 5-17-1 с tanh активацией и RBF с скрытым слоем из 7450 нейронов и гауссовой функцией активацией. Качество на тестовой выборке: R^2 и rmse(°C) для MLP равны 0.96 и 1.067, а для RBF - 0.95 и 1.12 соответственно.

Данные по нескольким городам - погодные данные, взятые из измерителей в аэропортах этих городов. Задача - предсказать температуру(или прочие погодные показатели, напр., влажность..) для города, не использованного в обучении, используя нейросетевые алгоритмы.

4 Получение данных

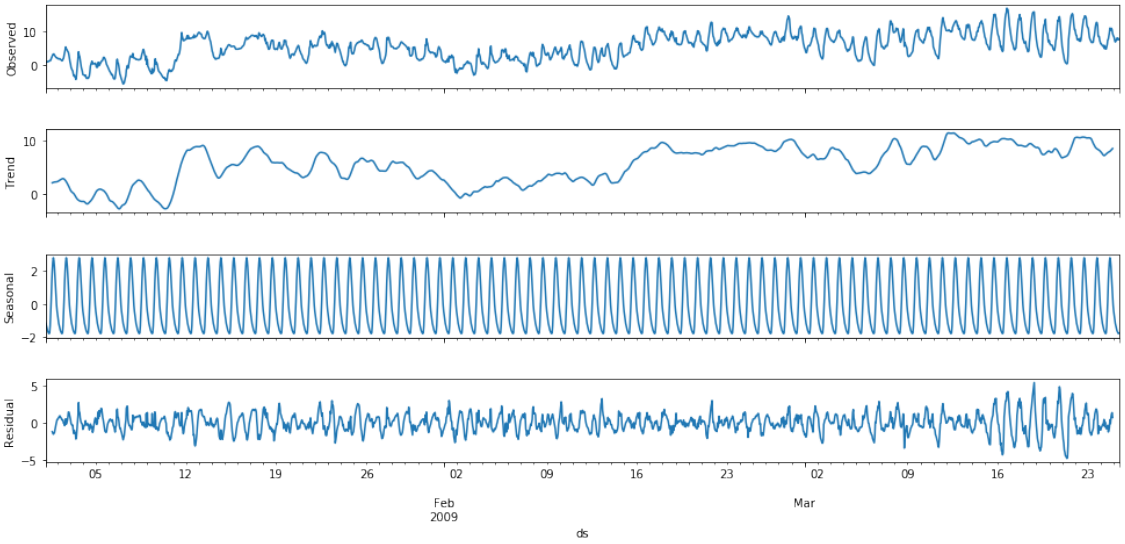
Данные были получены через API сервиса darksky и библиотеки для Python darkskylib. Для каждого города были скачаны почасовые данные(иногда с пропусками) с с 1 января 2009 по 1 января 2018 года. Следующие фичи были скачаны:

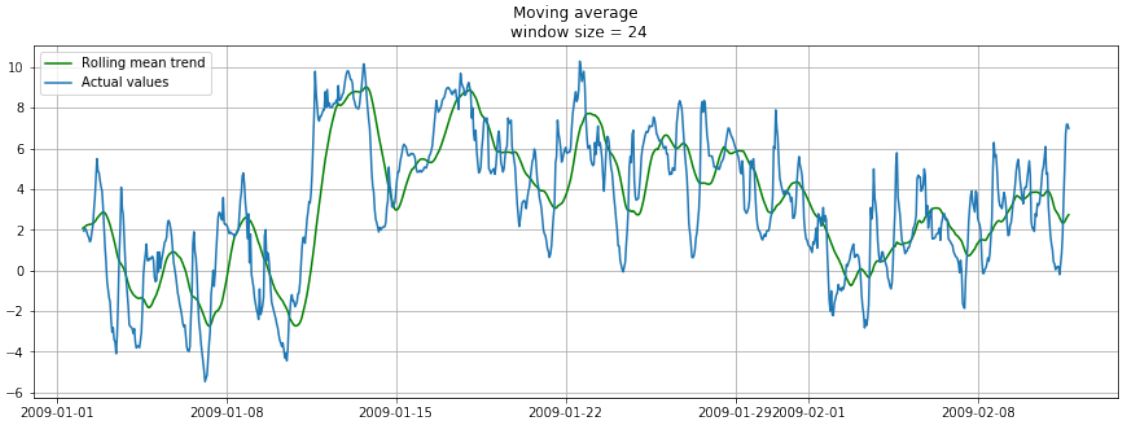
- температура воздуха(temperature)
- скорость ветра (windspeed)

- процент(от 0 до 1) покрытости неба облаками(cloudCover)
- относительная влажность, от 0 до 1 (humidity)
- видимость(visibility) - категориальная переменная, принимает только
- точка росы в градусах цельсия(dewPoint)
- краткое описание погоды - категориальная переменная (summary)
  - clear-day
  - clear-night
  - rain
  - snow
  - sleet
  - wind
  - fog
  - cloudy
  - partly-cloudy-day
  - partly-cloudy-night
- температура "по ощущениям" в градусах цельсия (apparentTemperature)

Алгоритм закачки данных: for каждый город в списке городов: for каждая дата в списке дат с 2009 по 2018 год с периодом в 1 час: скачать исторические данные за эту дату для этого города добавить данные за этот город в общий датафрейм

5 EDA





- Есть ненулевой тренд
- четко выраженная дневная сезонность
- годовая сезонность

6 Эксперименты

6.1 Схема валидации

Модели обучались на первых 80% данных - до 2016-03-15. Валидировались модели на оставшихся 20% данных - около 2х лет. Причина выбора такой схемы валидации проста - у нас имеется относительно большое кол-во данных(в сравнии с чем?) и более сложные схемы валидации, например, cross-validation on a rolling basis, оказываются не нужны для построения устойчивой оценки алгоритма. Более сложные схемы валидации часто применяют, когда данных мало и делить исходную выборку на 2 невыгодно.

6.2 Baseline

В качестве алгоритма для сравнения было взято простое предсказание температуры, равное предыдущему значению:

6.2.1 TODO написать в математической формулировке с формула как я предсказываю в baseline

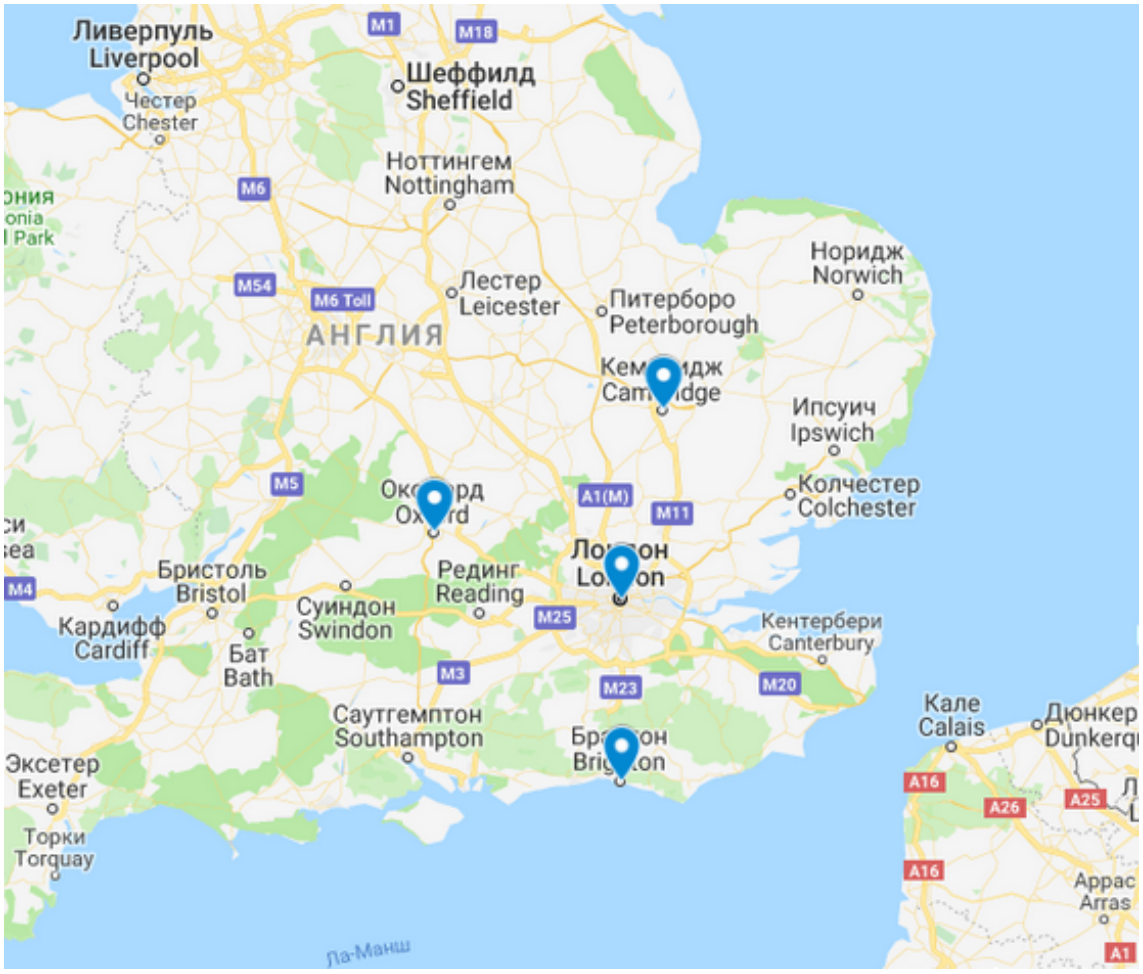
Этот алгоритм предсказывает на валидационной выборке с точностью до +/- 2.2 градуса Цельсия

6.3 MLP

Модель обучались на первых 80% данных - до 2016-03-15. Валидировалась - на оставшихся 20% данных - около 2х лет.

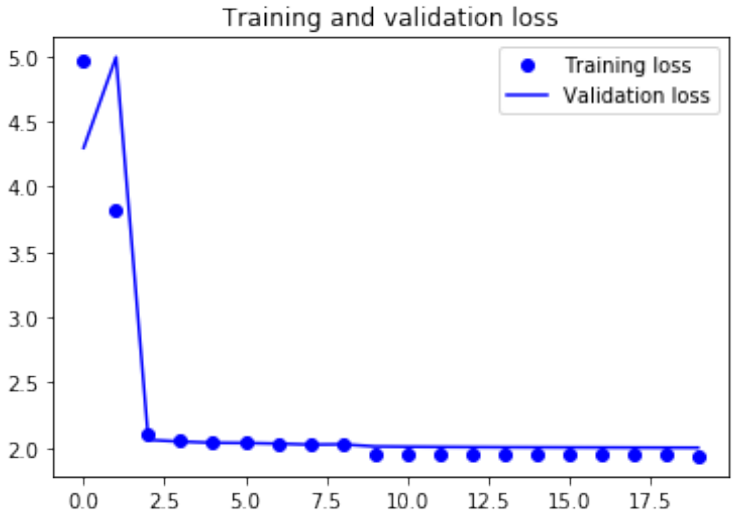
Данные были нормированы на среднее и дисперсию: $x_i = \frac{x_i - \overline{x}}{\sigma}$, где x - это отдельная фича или таргет (колонка в массиве объекты х фичи) и берется дисперсия и среднее этой фичи и она нормируется на свое среднее и свою дисперсию

фичи: только температуры 3 городов Оксфорд, Кембридж, Брайтон энд Хов, таргет - это Лондон.

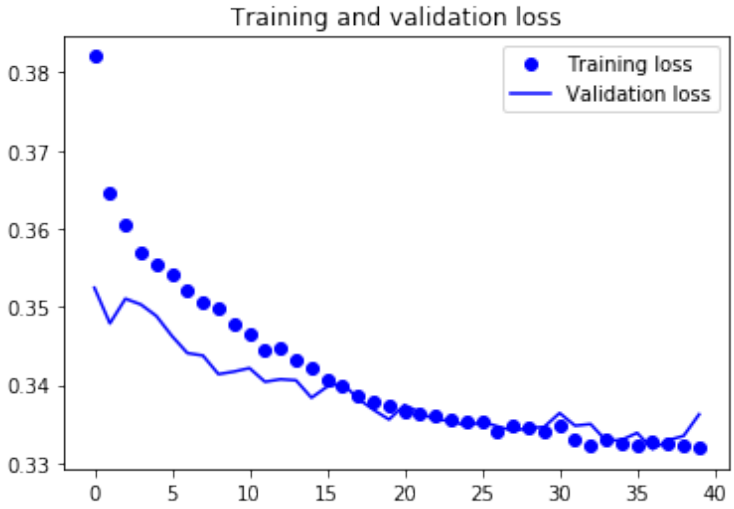


MLP обучался следующим образом: брали данные за 5 дней и температуру Лондона на следующие 24 часа. Оптимизировали mae. Архитектура нейросети: полносвязный слой с 32 нейронами и relu активацией и полносвязный слой с одним нейроном без функции активации на выходе. Оптимизатор - RMSprop. Для более быстрой и лучшей сходимости, скорость обучения делилась на 10, когда функция потерь на валидации увеличивалась или не изменялась:

Генератор данных на керасе для обучения нейросети был заимствован из книги "Deep learning with Python". См. код здесь



Отсюда видно, что нейросеть выучила всю информацию из данных и строить модель сильнее нет смысла. Например, если попробовать обучить LSTM на тех же данных, то можно увидеть, что сеть не сможет превзойти результат MLP:



6.4 LSTM

LSTM обучается абсолютно так же, как и MLP

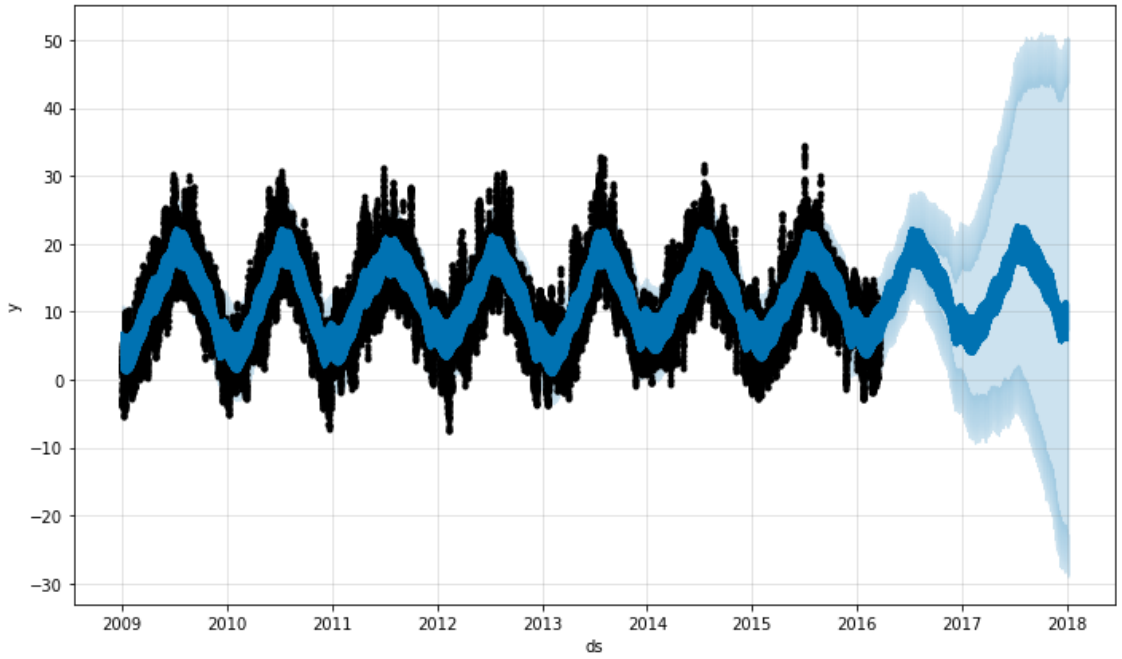
6.5 SARIMA(facebook prophet)

Так как наши данные периодичны с периодом в год, то вместо ARIMA, нужно использовать ARIMA с поддержкой сезонности - SARMIMA. SARIMA - это модель, которая обобщает линейную регрессию, всзвешенное усреднение, диференцирование временнного ряда, экспоненциальное сглаживание. Это все простые модели, которые можно проверить на наших данных, использовав только 1 модель - SARIMA.

SARIMA делает следующие допущения насчет данных - временной ряд стационарен:

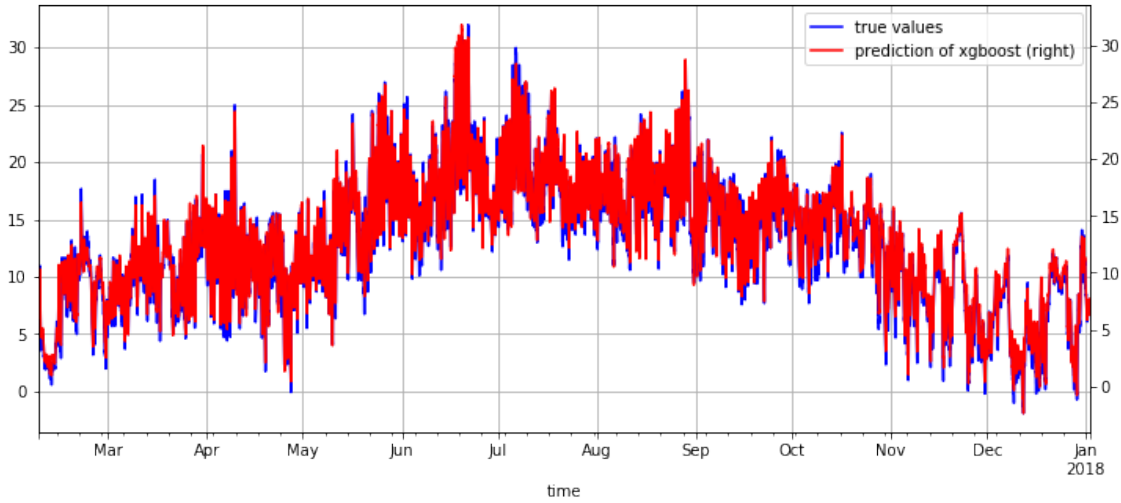
- нет тренда
- нет сезонности
- дисперсия всюду одинакова

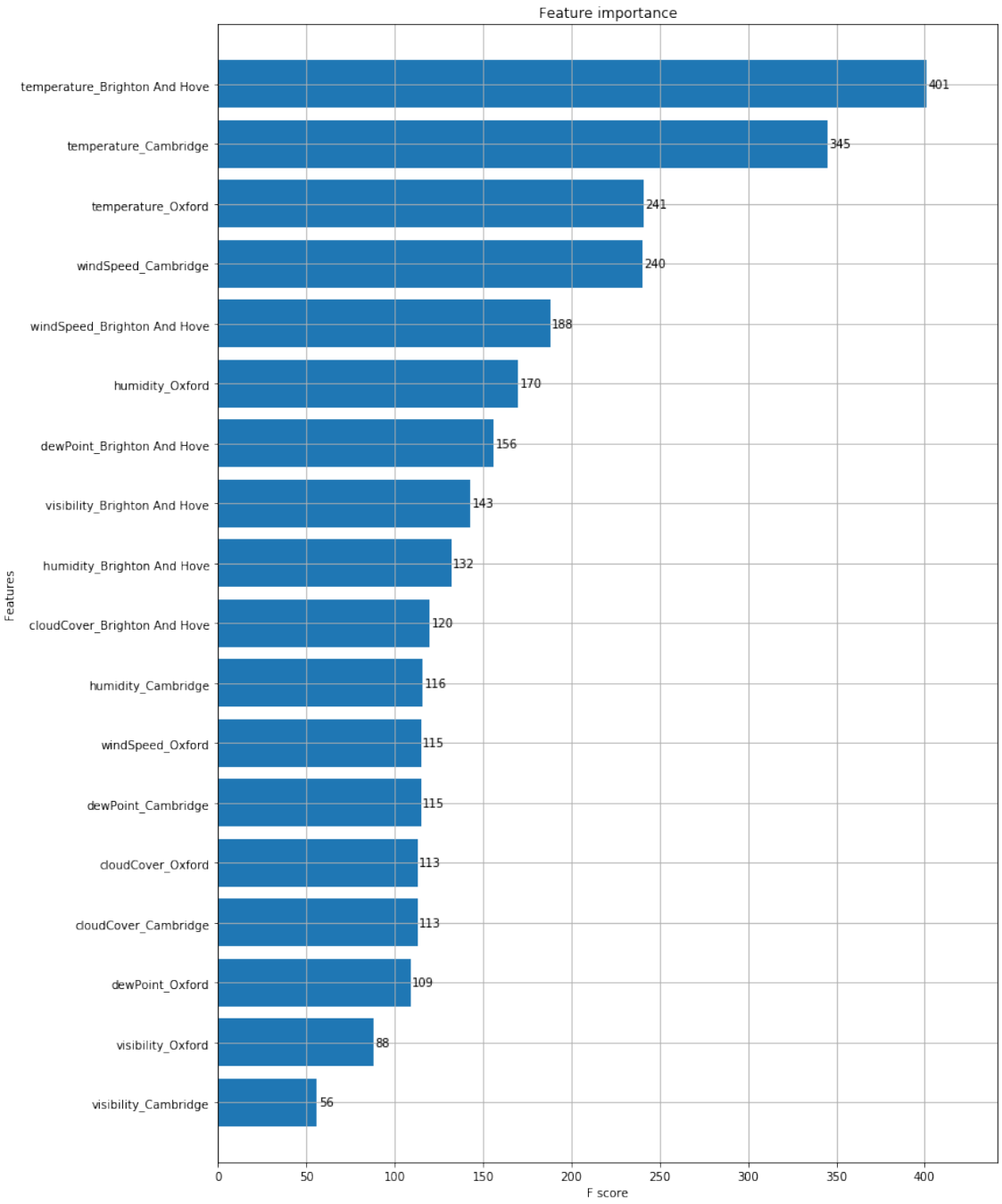
Проверку всех этих предположений, исправляение нестационарного ряда в стационарный и применененине SARMIMA реализовано в пакете facebook prophet



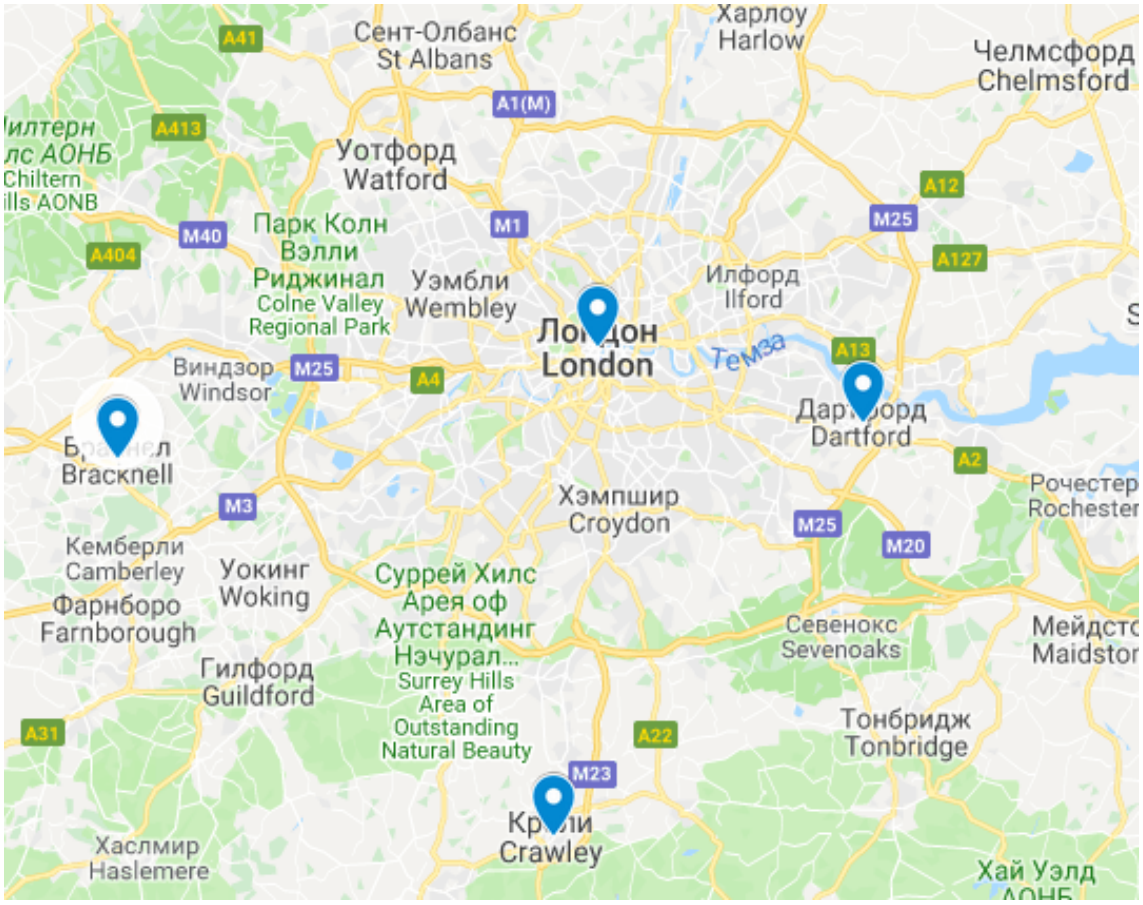
Из графиков видно, что fb prophet настраивается на тренд, но на колебания возле тренда настроится не может

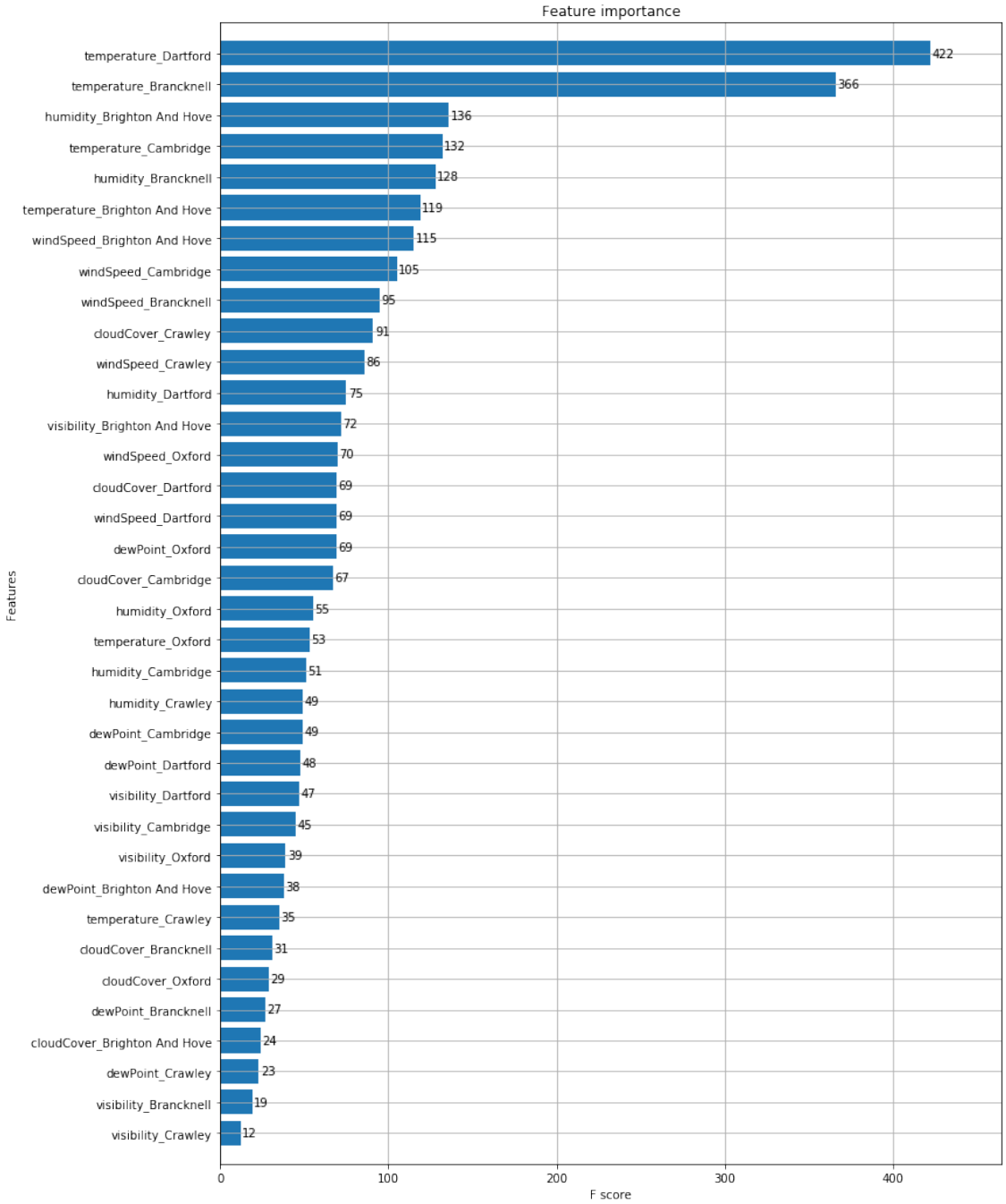
6.6 XGBoost

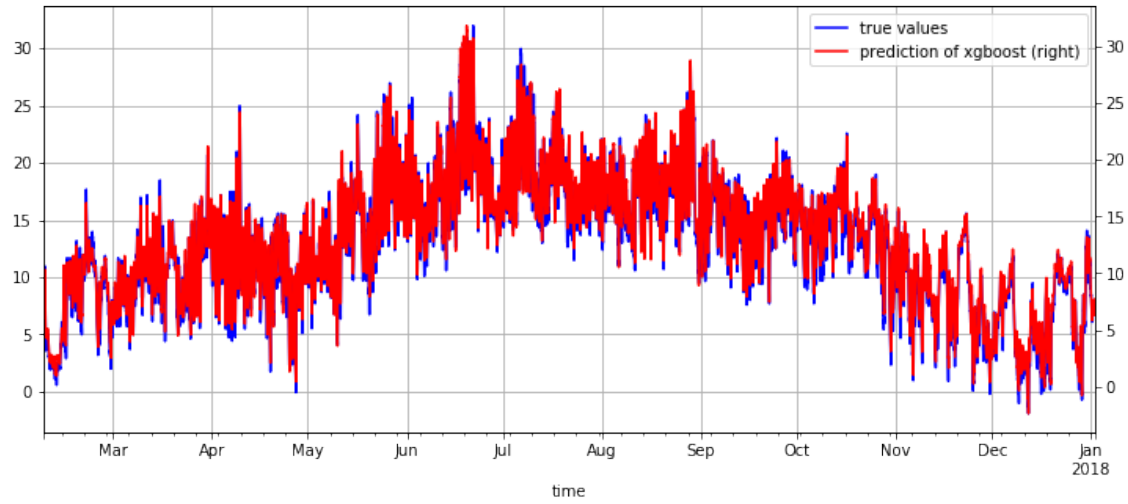

true values
prediction of xgboost (right)
30
25
20
15
10
5
0
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Jan
2018
time



После добавления еще 3х городов между Брайтон энд Хов и Лондоном


Сент-Олбанс
St Albans
Харлоу
Harlow
Челмсфорд
Chelmsford
A41
A1(M)
A413
Уотфорд
Watford
A12
M25
M40
M1
A404
A127
Парк Колн
Вэлли
Риджинал
Colne Valley
Regional Park
Уэмбли
Wembley
Илфорд
Ilford
Лондон
London
Темза
A13
Виндзор
Windsor
M25
A4
Дартфорд
Dartford
Бракнелл
Bracknell
A2
Рочестер
Rochester
M3
Хэмпшир
Croydon
M25
M20
Кемберли
Camberley
Уокинг
Woking
Суррей Хилс
Арея оф
Аутстандинг
Нэчурал...
Surrey Hills
Area of
Outstanding
Natural Beauty
Фарнборо
Farnborough
Севенокс
Sevenoaks
Гилфорд
Guildford
A31
A22
Тонбридж
Tonbridge
M23
Crawley
Хаслмир
Haslemere


Feature importance
temperature_Dartford
temperature_Brancknell
humidity_Brighton And Hove
temperature_Cambridge
humidity_Brancknell
temperature_Brighton And Hove
windSpeed_Brighton And Hove
windSpeed_Cambridge
windSpeed_Brancknell
cloudCover_Crawley
windSpeed_Crawley
humidity_Dartford
visibility_Brighton And Hove
windSpeed_Oxford
cloudCover_Dartford
windSpeed_Dartford
dewPoint_Oxford
cloudCover_Cambridge
humidity_Oxford
temperature_Oxford
humidity_Cambridge
humidity_Crawley
dewPoint_Cambridge
dewPoint_Dartford
visibility_Dartford
visibility_Cambridge
visibility_Oxford
dewPoint_Brighton And Hove
temperature_Crawley
cloudCover_Brancknell
cloudCover_Oxford
dewPoint_Brancknell
cloudCover_Brighton And Hove
dewPoint_Crawley
visibility_Brancknell
visibility_Crawley
422
366
136
132
128
119
115
105
95
91
86
75
72
70
69
69
69
67
55
53
51
49
49
48
47
45
39
38
35
31
29
27
24
23
19
12
Features
0
100
200
300
400
F score



6.7 Результаты экспериментов

baseline модель предсказывает температуру на следующий час по предыдущему значению, для нее нет смысла в разделении выборки на обучающую и тренировочную

модель	mae на валидационной выборке, градусы цельсия	mae на обучающей вы
baseline	2.20	-
MLP	2.00	
LSTM	1.99	
XGBoost, 6 городов	0.18	0.176
XGBoost, 3 города	0.46	0.427
SARIMA(fb prophet)	9.2	

7 Ссылки

Основные понятия и обозначения в машинном обучении. Воронцов К.В.

8 Список литературы

References

[1] Seth E. Snell, Sucharita Gopal, and Robert K. Kaufmann. Spatial interpolation of surface air temperatures using artificial neural net-

works: Evaluating their use for downscaling gcms. *Journal of Climate*, 13(5):886–895, 2000.

[2] Imran Tasadduq, Shafiqur Rehman, and Khaled Bubshait. Application of neural networks for the prediction of hourly mean surface temperatures in saudi arabia. *Renewable Energy*, 25(4):545 – 554, 2002.

9 Приложение

9.1 Алгоритм закачки данных с darksky

```
cities = get_cities(['Oxford', 'Cambridge', 'Brighton And Hove', 'London'])
for city, key in zip(cities[: len(keys)], keys.keys()):
    df = pd.DataFrame()
    date_start = get_last_downloaded_date(city[0])
    date_end = dt(2018, 1, 1, hour=0)

    try:
        date_list = get_list_of_days(date_start, date_end)
    except AssertionError:
        continue

    logger.info(f"Скачиваю данные с города {city[0]}")
    for date in tqdm(date_list):
        try:
            _city = forecast(key, city[1], city[2], time=date)
            error = False
        except requests.exceptions.HTTPError as e:
            error = True
            logger.error(str(e.request) + str(e.response) + str(e))
            break
        except Exception as e:
            error = True
            logger.error(str(e))
            break
        try:
            for i in range(len(_city.hourly)):
                values = [to_date_from_unix_time(_city.hourly[i]['time'])]
                for column in columns:
```

```
                    try:
                        values.append(_city.hourly[i][column])
                    except KeyError as e:
                        values.append(None)
                    t = pd.DataFrame(values).T

                    df = pd.concat((df ,t))
                except AttributeError as e:
                    logger.error(str(e))
                    error =True

            if df.shpape[0] > 0:
                df.columns = ["time"] + columns
                df = df.set_index("time")

                path = os.path.join(CURRENT_DIR, "diplom_data/" + str(city[0]) + ".csv")
                with open(path, 'a') as f:
                    df.to_csv(f, index=True, header=False)

                    keys[key] = True #key is used, dont use it again today
```

9.2 Код самого первого решения (baseline)

```
def evaluate_naive_method():
    batch_maes = []
    for step in range(val_steps):
        samples, targets = next(val_gen)
        preds = samples[:, -1, 1]
        mae = np.mean(np.abs(preds - targets))
        batch_maes.append(mae)
    return np.mean(batch_maes)
```

9.3 MLP

```
model = Sequential()
model.add(layers.Flatten(input_shape=(lookback // step, data.shape[-1])))
model.add(layers.Dense(32, activation='relu'))
model.add(layers.Dense(1))

model.compile(optimizer=RMSprop(), loss='mae')
```

```
callbacks_list = [
    keras.callbacks.ReduceLROnPlateau(
    monitor='val_loss',
    factor=0.1,
    patience=1,
    verbose = 1
    )
]
history = model.fit_generator(train_gen,
                              steps_per_epoch=train_steps,
                              epochs=20,
                              validation_data=val_gen,
                              validation_steps=val_steps,
                              callbacks=callbacks_list)
```