Краткая формулировка задачи.

Задача состояла в предсказывании цены на товар item_id за год year и неделю week. В роле фичей выступали продажи за какое-то предыдущее время. Параметр shift говорил о том, начиная с какой даты нам давали данные о продажах. К примеру, данные с shift == 3 за (year, week) и данные с shift == 2 за тот же (year, week) отличались сдвигом на 1 фичу "в прошлое".

Описание итогового решения: как готовились данные, что использовалось в качестве таргета, какой алгоритм обучался и все комментарии, которые могут быть полезны для воспроизведения вашего решения.

Итоговое решение включает в себя 3 действия:

1. Работа над данными

— Выкинуть шифты 2, 3 и оставить только shift==1, потом удалить этот столбец за ненадобностью.
— Выкинуть те item_id, которых нет в тесте (да, это могло лишить нас понимания характера зависимости, но это улучшало ошибку на CV)
— Выкинуть f31 - f60. Построив график, я убедился, что это копирование первых 30 фичей.
— Бинаризировать параметр item_id, т.к. это категориальный признак с кучей значений.
— Посортить train_data для правильной кросс валидации.
— Заметить, что если взять одинаковый item_id и посмотреть на какую-то неделю и следующую за ней, то получится, что $y_{current}$ зависит от $f30_{next}$ линейно для каждого item_id. Увидеть это можно просто построив график y и какой-то фичи, получится, что выглядят они одинаково, однако у сдвинут по оси x и y.
— После вышестоящего замечания, можно получить примерные ответы на 2/3 выборки. Я взял примерную константу 0.6211, полученную путём взятия np.diff и деления np.diff(y) на np.diff(f30), но можно также натренировать какой-то алгоритм, но к этому времени я уже слишком устал от контеста, поэтому взял константу.

2. Модель

— Возьмём холдаут в 10% для независимого чека нашего решения
— Построить нормальную CV. Для этого разделим данные на k частей и будем трейнится и тестится так:
[1] --- [2]
[1][2] --- [3]
[1][2][3] --- [4] и т.д.
— По CV определился фаворит - ExtraTreesRegressor, проверим его на холдауте - ошибка адекватная, затюним его параметры

3. Грязный хак

— Теперь подставим наши читерные ответы на те места, на которые можем, на все остальные поставим те ответы, которые нам предсказал алгоритм.

Зашлём ответ в контест.

Рассказ о подходах, которые вы пробовали, и том, как реализация тех или иных идей сказывалась на качестве вашего решения.

В начале я попробовал втупую построить более крутой алгоритм на 100% данных.
Появилась проблема - как проверить ответ. Вспомнив лекцию, я написал свою кросс валидацию для даты, которая зависит от времени.
Далее, потратив попыток 5, я понял, что public leaderboard score ниже 25 у меня не станет.

Надо было двигаться дальше, я решил посмотреть графики фичей. Сразу увидел, что f1-f30 == f31-f60, выкинул их.

Была также идея с шифтами, т.е. по сути 29 полей между shift == 1 и shift == 2 совпадают, но одно появляется новое, я решил вычленить их. Потратя 10 часов кодинга и попыток вытащить всё это дела из pandas, я это сделал, но в итоге в конце не использовал (хотя, может, это дало бы что-то, т.к. появились 2 лишние фичи), но от контеста я уже очень устал в конце и мой запал угас [1].

В конце концов, я построил график первой фичи и y для item_id, получилось очень забавная штука - y это сдвинутый f1 (для любого f_x это работало, как оказалось потом). Я попытался подогнать константу и реально получались очень маленькие ошибки. Тут встал вопрос - вбить ли ответ ручками или научить какой-то алгоритм на этом. Логичнее было научить алгоритм, потому что константа была неочень точная, но опять же мой запал угас [2].

Попробовав несколько алгоритмов, я отослал самый лучший по CV и всё.

Основное время занял, конечно, дата майнинг. Моё решение можно было затюнить и получить что-то лучше моего скора, но мой запал угас [3].

Код вашего итогового решения.

Код приложен файлом solution.ipynb

Описание того, как вы оценивали качество при решении контеста: как делали кросс-валидацию, насколько она коррелировала с результатом по leaderboard.

Про кросс-валидацию было написано выше. Коррелировало ли CV с результатом по leaderboard? Несомненно. Обычно, чем выше скор на CV, тем лучше по leaderboard. Однако если скор на CV отличается совсем не намного, то это не гарантируется, конечно. Ниже код CV. Он несколько сложен, т.к. я сделал blending для кучи алгоритмов со взятием среднего :)

In [1]:
```
def cross_val_time_dependent(size, k=5):
    """
    [1] - [2]
    [1][2] - [3]
    [1][2][3] - [4]
    [1][2][3][4] - [5]
    """
    parts = [0] + [size / k] * (k - 1) + [size / k + size % k]
    parts = np.cumsum(parts)
    return np.array([(np.arange(parts[pc - 1]), np.arange(parts[pc - 1], parts[pc]))
                     for pc in np.arange(1, k + 1)])[1:]
```

In [2]:
```
def perform_cros_val(estimators, folds, fit_data=train_data,
                     fit_labels=train_labels, o=False):
    err = 0
    counter = 1
    err = np.zeros(folds - 1)

    for cv in cross_val_time_dependent(train_data.shape[0], folds):
        train_indices, test_indices = cv
        for estimator in estimators:
            estimator.fit(train_data.iloc[train_indices],
                          train_labels.iloc[train_indices].values)
        # Трейним кучу алгоритмов

        mean_predicted = np.array(estimators[0].predict(train_data.iloc[test_indices]))

        for i in np.arange(1, len(estimators)):
            mean_predicted = np.vstack((mean_predicted,
                                        estimators[i].predict(train_data.iloc[test_indices])))
        # берём среднее ошибки между ними
        if len(mean_predicted.shape) > 1:
            err[counter - 1] = get_sMAPEError(train_labels.iloc[test_indices].values,
                                              np.mean(mean_predicted, axis=0))
        else:
            err[counter - 1] = get_sMAPEError(train_labels.iloc[test_indices].values,
                                              mean_predicted)
        if o:
            print "did %d folds" % counter
            # апут, чтобы знать, сколько ещё ждать-то
        counter += 1

    return (np.sum(err) / (folds - 1), np.max(err), )
```

...

Дополнительные задачи.

1. sample_submission.tsv сформирован как среднее фич f1-f60