

Отчёт по задаче «Предсказание затрат»

Каюмов Эмиль, 517
ММП ВМК МГУ

Курс «Прикладные задачи анализа данных»

Задача

- По истории покупок необходимо предсказать сумму первой покупки на следующей неделе
- Дата и сумма за 1.5 года
- Если клиент ни разу не придёт в магазин в течение недели, то правильный ответ 0
- Метрика: classification accuracy (17 классов)

Мысли и замечания

- Клиент может ходить реже 1 раза в неделю, поэтому не попадёт в «нужную» неделю
- Клиент может уже давно не ходить в «наш» магазин
- Accuracy, то есть нас не интересуют различные естественные усреднения суммы, а только конкретное значение из 17 возможных
- Выборка не содержит 0, то есть их надо делать отдельно или обойтись без них, предсказывая отдельно сумму и приход.

Валидация

- Оценка качества производится на последней неделе, валидацию сделаем таким же образом:
 - Отрежем последнюю неделю данных
 - Возьмём первую покупку каждого клиента
 - Для клиентов без покупки поставим 0
- Сделаем так для 7 последних недель и усредним
- Std качества предсказания около 0.006, но вполне соответствует лидерборду (с точностью до 3 сабмитов...)

Простое решение

- Невыгодно усреднять — возьмём моду
- Занулим «покинувших», если давно (по $cv = 7$) не ходили
- Занулим тех, кто в среднем ходит с такой частотой, что не придёт в течение «нужной» недели (по $cv = 7$)
- Занулим *зачем-то* тех, кто в среднем ходит с такой частотой, что придёт до «нужной» недели (по $cv = 4$)

Простое решение

```
f_delta = -4
f_minus = -7
f_plus = 7
last_day = 438

data = pd.read_csv('data.csv')
data['week'] = data['date'].apply(lambda x: int(x.strftime('%W')))

train_gr = train.groupby('id')
tmp_prediction = train_gr.apply(lambda x: mode(x)[0][0]).reset_index().set_index('id')

prediction = pd.DataFrame({'id': np.unique(data.id)})
prediction['sum'] = prediction.id.map(tmp_prediction['sum']).fillna(0).astype(int)








train['delta_day'] = train['date'].apply(lambda x: (x - train['date'].min()).days)
prediction['delta_day'] = prediction.id.map(train_gr['delta_day'].mean())
prediction['last_day'] = train_gr['last_day'].reset_index(drop=True)['date']

def make_zero(row):
    if row['last_day'] < f_minus \
    or row['delta_day'] - last_day < f_delta \
    or row['last_day'] + row['delta_day'] - last_day > f_plus:
        return 0
    else:
        return int(row['sum'])

prediction['sum'] = prediction.apply(make_zero, axis=1)
prediction[['id', 'sum']].to_csv('../output/submission_3.csv', index=False)
```

- LB: 0.38139
- Второе место на момент сабмита!

Ничего не предвещало беды перед дедлайном...

12	▲ 4	Yura Trubitsyn (MMP, MSU, R...		0.39290	9	1d
13	▲ 1	Anna Pidzhakova(MMP, MSU, ...		0.39254	8	4d
 dummy_benchmark.csv				0.38357		
14	▼ 5	Artem Popov (MMP, MSU, Rus...		0.38333	16	3h
15	▼ 4	Emil Kayumov (MMP, MSU, R...		0.38139	2	12d
<div><div>Your Best Entry ↑</div><div>Your submission scored 0.38139, which is an improvement of your previous score of 0.37587. Great job!</div><div> Tweet this!</div></div>						
16	▼ 4	Eugen Bobrov (MMP, MSU, Ru...		0.37672	7	18h



Blending, stacking or one more thing...

- Раз уж кода мало, действуем по минимуму, ведь терять нечего
- Добавим веса к сумме покупок (по дням так, чтобы последний день соответствовал 1)
- Немного подберём степень ($cv = 1.2$)
- LB: 0.38533

Выводы

- NO lstm
- NO xgboost
- NO stacking
- YES штраф

P.S. Стоит уточнять точные критерии заранее, а не рассчитывать на логику...

Let's play...

