飯田産業 土地の販売価格の推定
3rd solution

taksai

自己紹介

齊藤　拓磨　(taksai)

株式会社　カカクコム
マーケティング支援室 データサイエンスチーム リーダー

価格.com、食べログを中心に様々なwebサイトのデータサイエンスを担当
特にログデータやDBデータの、分析・機械学習タスクが多い

SIGNATEはほぼ初めの参加、kaggleは今年始めてまだ銅1個（これから頑張る）
競馬AI一緒にやりたい人、データサイエンス友達募集中！

全体的な感想

- 建物有無の判別が難しかった
- データ量はさほど多くなかったので、色々と試行錯誤できた
  - 結構時間使った。。
- 説明変数予測したものや、目的変数変えたものを、色々混ぜて精度向上できた

- 直前はダメだと思った (可愛いくせに強い女は...)
- 優勝したかった

取組概要

- データ加工
- 説明変数の予測
- 目的変数
- 学習
- アンサンブル

取組概要

- データ加工
- 説明変数の予測
- 目的変数
- 学習
- アンサンブル

データ加工

- 住所のクレンジング（繰り返し、字以降削除）
- keiyaku_prが明らかに1/10程度の3レコードを10倍に
- 土地建物フラグを作成
  - tateuri_su、 tochiuri_su、 tt_mseki、levelplan、hw_statusより判断
  - 私には良く分からないものは後ほど予測

- 土地建物フラグをベースにlevelplanを修正
  - 土地っぽかったら「土地売り」など

- いくつかの変数はダミー化
  - hokakisei、kobetsu、setsudo_hi、jukyo、levelplanなど

データ加工

- eki_kyori_tohoをbas_toho毎に分割
  - バスは徒歩の4倍、車は徒歩の5倍早いとした
- 住所に対して緯度経度を付与
- 土地価格の変数を作成
  - rosenka * tc_msekiなど

一番力を入れたのは、どれが建物なのかという判断
予測結果を見てどう予測を外しているのか分析しながら試行錯誤した

データ加工

データ加工前後のlocalCVは以下
- データ加工前：9.54
- データ加工後：8.54

精度が上がらないデータ加工も含めて様々試したが、
データ加工はやっぱり重要

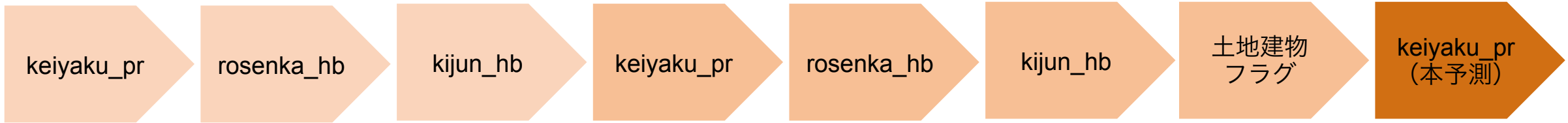
取組概要

- データ加工
- 説明変数の予測
- 目的変数
- 学習
- アンサンブル

説明変数の予測

- ３つの説明変数を予測
  - rosenka_hb、kijun_hb：
    - 0以外のデータを学習データとして、0のデータを予測して埋めている
  - 土地建物フラグ（自作、0 or 1のデータ）：
    - 私が判断できたものを学習データとして、不明だったものを0~1で埋めている
- 上記説明変数を予測する際、keiyaku_prも説明変数に入れている
- testデータも入れており、そのkeiyaku_prは予測したものを使用

- 以下のような流れで説明変数を予測をした後に本学習へ
  - 以下、何を目的変数としているか記載



説明変数の予測

説明変数予測前後のlocalCV[mape]は以下

- データ加工前：8.54
- データ加工後：8.24

rosenka_hbやkijun_hbは土地価格を決める際の重要な変数
これらと建物土地フラグを予測することで大幅に精度が向上できた

取組概要

- データ加工
- 説明変数の予測
- 目的変数
- 学習
- アンサンブル

目的変数

以下の目的変数を予測してアンサンブルしている

- keiyaku_pr
  - log1pをとって、metric：rmseで学習
    →rmseで効率的に学習させながらmapeを最適化可能
  - 単体localCV：8.2程度

- keiyaku_pr – 土地価格
  - 建物価格予測みたいなタスクに
  - 単体localCV：8.4程度

- keiyaku_pr – (土地価格 + tt_mseki * 200,000)
  - 最終項は建物価格っぽくして、土地 + 建物価格とkeiyaku_prの差を予測
  - 単体localCV：8.4程度

取組概要

- データ加工
- 説明変数の予測
- 目的変数
- 学習
- アンサンブル

学習

以下アルゴリズムで、fold8~11辺りを使用して学習
目的変数は基本keiyaku_pr、lightGBMでは前頁記載の他も使用
汎化性能を高めるため、最終的には22個のモデルを作成

- 使用したアルゴリズム（数字は単体精度[mape]）
  - lightGBM（gbdt：8.2, dart：8.5）
  - XGBoost（gpu_hist：8.3）
  - Keras（9.7、dropout有り[2048, 512, 128, 32, 1]程度）
  - Pytorch（9.4 、dropout有り[9064 , 2048, 512, 128, 32, 1]程度）

取組概要

- データ加工
- 説明変数の予測
- 目的変数
- 学習
- アンサンブル

アンサンブル

作成した22個のアルゴリズムで単回帰
→汎化性能が高まるように、同じぐらいずつモデルを使用したいため、
　手で配合を調整

最終的には1~10%ずつ入れるような配合がsubmit時に最高に
→これでlocal：8.06、暫定評価：8.20、最終評価：8.259

単体ではlocal：8.2程度がmaxだったので、
アンサンブルにより汎化性能が向上

コンペを終えて

一番効いたのはデータ加工

更に目的変数の加工や、説明変数の予測で精度を向上

精度向上に向けて

もう少しNNの精度を高められたら

優勝を狙っていたが、当日は厳しいかと思った...（直前7位ぐらいだった）

諦めずに出来ることを最後までやりきったのが3位に繋がった

次回は優勝できるように！