特徴量とかモデルの管理とかの思索

threecourse

Kaggle Meetup 2016/3/5

プロフィール

threecourse

アクチュアリー（生命保険）

- 仕事で機械学習使わない
- というか Excel と VBA がメイン。勝手に C#とか Python を使っている。

Kaggle とのお付き合い

- 初めて参戦した 2015/5 の walmart recruiting ii で優勝
  - 決まり手は曲線フィッティング＋差分を vowpal wabbit で linear regression
  - 与えられた天候データが壊れていて、コンペで通常は有効な手法が通用しづらかったと思われる
- さすがに楽しくなって kaggle 歴 1 年
- kaggle に勝つための知識に絞って学んでいます
  - 社会人は時間がない（学生も？）
  - 理論的な側面はかなり弱いので、かなり素人な質問をすると思います
  - Solution/Forum は頑張って読んでます

なぜ Kaggle をやるか

- 自己顕示欲
- 楽しさ (with 苦しさ、寝不足)
- キャリアの幅を広げる
- スキルアップ

問題意識

- コンペを進めていくうちに、コードが複雑化していってしんどくなる
  - 仕事じゃないので、しんどいとつらい
- 細かいところで特殊対応をすると、そこを修正するコストが結構ある
- できるだけ自動化できるところは自動化したい
  - パラメータ調整、アンサンブルや特徴量選択など
- Walmart 2015, Prudential はこの辺の管理について考えながらやっていた
- コードはこちらにあります
  https://github.com/threecourse/kaggle-prudential-sample

問題意識 (cont)

問題となりやすいのは、特徴量とモデル

- 特徴量
  - 特徴量を思いついて追加したり、xgboost 用と nn 用と vowpal wabbit 用の feature set をそれぞれ作ったりすると、 わけが分からなくなる
- モデル、アンサンブル
  - ツールごとに微妙にインターフェースが違うが、適当に作っていくと、cv のやり方を変えたり、hyperopt したりするときにそれぞれ細かいところを変えなくてはならなくなってしまう

特徴量についての要件

- 特徴量は思いつくままに作って、最後に取捨選択したい
- 必要な場合は CVFold ごとに特徴量を作れるようにしたい
  - 少なくとも、教師データを使う特徴量（例：Coupon では、ある条件のときの購入確率）を作るときなどはこうしないとリークしてしまう
  - Chenglong Chan, 4.1.2 Following the Same Logic 参照
  - TF-IDF 等であれば全体で作ってしまっても良いような気もしますが
- LabelEncoder や TF-IDF などを考えると、train data と validation/test data を同時に処理したい
- 作った特徴量への変換 (ダミー変数にするとか）を柔軟に適用したい
- Stacking のために、計算結果を再度特徴量として読み込めるようにしたい

特徴量管理の実装

特徴量の表現

特徴量は pd.DataFrame として表現する

- index をレコードのユニークな id とする。reindex することで、各 fold のデータを取り出す
- columns は１つだけのことも、複数のこともある
- ある程度細かく分けた方が、特徴量選択には使いやすそう

特徴量管理の実装 (cont)

feature_set の表現

feature_set を (タグ, feature 名のリスト, 変換手法のリスト) のリストとして表現する
例 を見ていただいた方が良いかも

- ユーティリティ関数を作ってこれを処理する。
  - feature を読み込み、前処理を適用し、hstack で結合し、train/test に分ける
- タグは keras graph や vowpal wabbit で使用できる（あまり効果は無いかも）
- ちょっと冗長な気がしないこともない

特徴量管理の実装 (cont)

保存場所

特徴量は以下に保存する

- feat/common に CVFold で共通の特徴量
- feat/fold0, fold1.., feat/all に CVFold ごとに作成する特徴量

なお、モデルの出力結果は以下に保存する

- model/<model name>/fold0, fold1..., model/<model name>/all
- stacking に使うモデルの計算結果もこのフォルダを活用

モデルについての要件

- いろんな feature_set で試したい
- hyperopt を使いたい
- xgboost にしても NN にしても、early stopping を使いたい
- 結構微妙にインターフェイスが違う（特に vowpal wabbit...）
- 基本的には、test data の予測値とモデル自体を返せば良い。
  - モデルはそんなに使うことはないが、feature importance を見るためなどに使用する

モデルの実装

- ある model を扱うファイルには以下を入れておく
  - run_model メソッド
  - （run_model_hopt メソッド）
  - Model クラス（xgboost や keras の model を wrap する）
- 柔軟性を持たせるためにこのようにしている
- 微妙に各ツールの model のインターフェイスが違うので wrap している

モデルの実装 (cont)

run_model メソッド

- 引数
  - モデル名・パラメータ名・feature_set 名の tuple からなる ModelSetting クラス
  - i_fold(cv_fold の何番目か)
- 計算の流れは、
  1. parameter のセット
  2. data の読込・セット
  3. train
  4. predict
- 返り値は test data の予測値（とモデル）
- hyperopt ではパラメータを動かすので、また少しインターフェイスが異なってくる

モデルの実装 (cont)

Model クラス

- 保持するデータ
  - label(train/test) - 目的変数
  - data(train/test) - 特徴量
  - parameter
  - model - xgboost や keras の model
- kaggle では early stopping などのために validation set を一緒に持っておくことがほとんどなので、最初からそれを前提にする

モデルのラン

- モデル, feature_set, パラメータを指定すると、ランを行える
- hyperopt でパラメータサーチからの一番良いパラメータでランという流れを実装した
  - overfit しないよう、hyperopt のためには別の cvfold を作成した
  - 上位のパラメータから適度に diverse なものを選ぶような方法もありそう
- モデルの出力結果を feature として取り込めるようにしておけば、stacking はすっきりと書ける