Bike Share 業務改善提案 発表

2017/08/03

hidecha

課題仮説設定と解決のアプローチ

課題

- 自転車の貸出と返却の行動パターンを分析し、自転車とドックの供給不足を可能な限り防ぐ

仮説

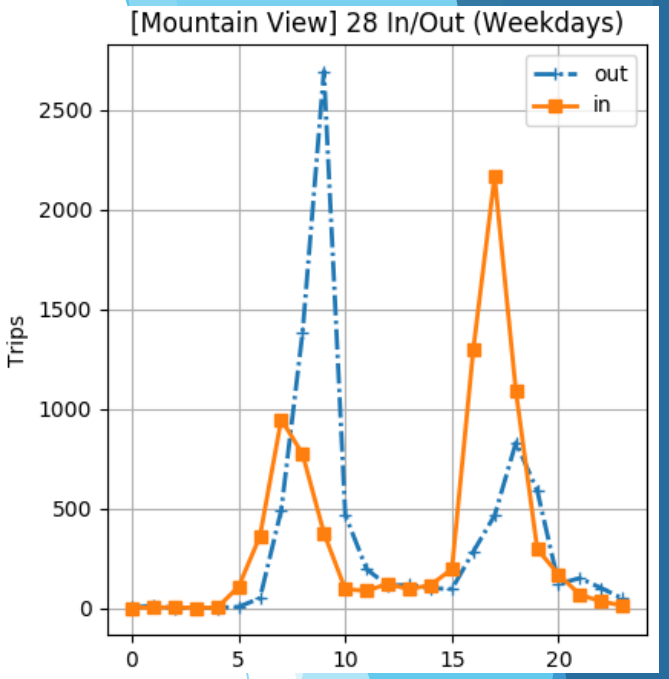
- 大学や駅周辺など時間帯により貸出、返却が集中すると供給不足を招くのでは？
- 他にも例えば、標高が高い場所で借りて低い場所で返すユーザーが多いと偏りが生じるのでは？

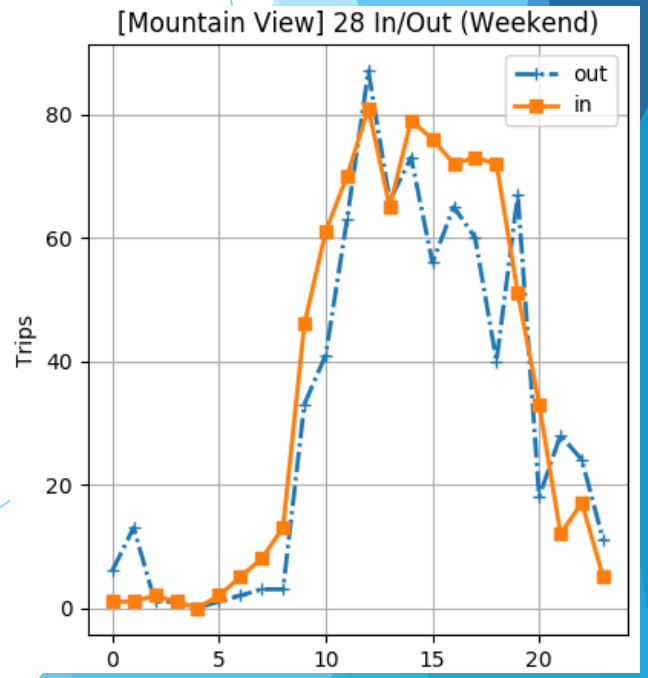
課題解決のアプローチ

- ステーションごとに貸出、返却の傾向を分析し、貸出数と返却数の予測を行う
- パラメーターは曜日、年初からの週、時間、分、天気などを検証する
- 偏りを平準化する施策を考案する

Tripデータを元にした現状分析

- 都市 (San Jose, San Francisco, Palo Alto, Mountain View) ごとにTripデータを分け、Start Terminal / End Terminal を軸にデータを加工
  - ✓ Redwood City は2016年のstationデータで存在しないため調査対象外とする
- 3年分のデータを、ステーションごとに分け、Start (=Out) / End (=In) のTrip数を平日 / 週末に分け、かつ時間ごとに集計しグラフ化
- このグラフにより、In / Out のギャップが大きい時間帯で供給不足が想定される
  - In >> Out: ドックが不足しやすい
  - In << Out: 自転車が不足しやすい
- 右図はステーション**28: Mountain View Caltrain Station** における平日(上図)、週末(下図) のTrip数の時間ごとの集計
  - 平日7-10時頃は In << Out のため自転車不足、
  - 平日16-19時頃は In >> Out のためドック不足と推測される
  - 週末は平日に比べてTrip総数が少なく、In / Out のギャップは大きくないため、供給不足は発生しにくいと推測される





Source code: https://github.com/hidecha/Bike/blob/master/trip_delta_upd.py
https://github.com/hidecha/Bike/blob/master/trip_delta_plt.py

Statusデータを元にした現状分析1

- Statusデータをステーションごとに bikes_available=0, docks_available=0 となった回数を集計し、多い順にソート (TOP5)

	bikes_available=0		docks_available=0	
	station_id	count	station_id	count
San Jose	4	14,307	4	12,429
	10	11,641	7	11,053
	5	7,080	84	6,801
	2	5,416	3	6,261
	6	5,140	9	5,376
San Francisco	70	43,079	62	44,844
	54	39,401	45	44,728
	73	33,112	48	35,903
	60	25,476	60	32,980
	69	23,605	41	32,505
Palo Alto	37	7,726	35	12,212
	34	4,202	37	3,604
	35	2,344	38	2,641
	36	79	36	2,161
	38	65	34	0
Mountain View	32	20,712	27	5,658
	27	15,151	32	2,022
	28	4,077	30	1,713
	31	4,036	28	1,232
	29	2,644	31	1,037

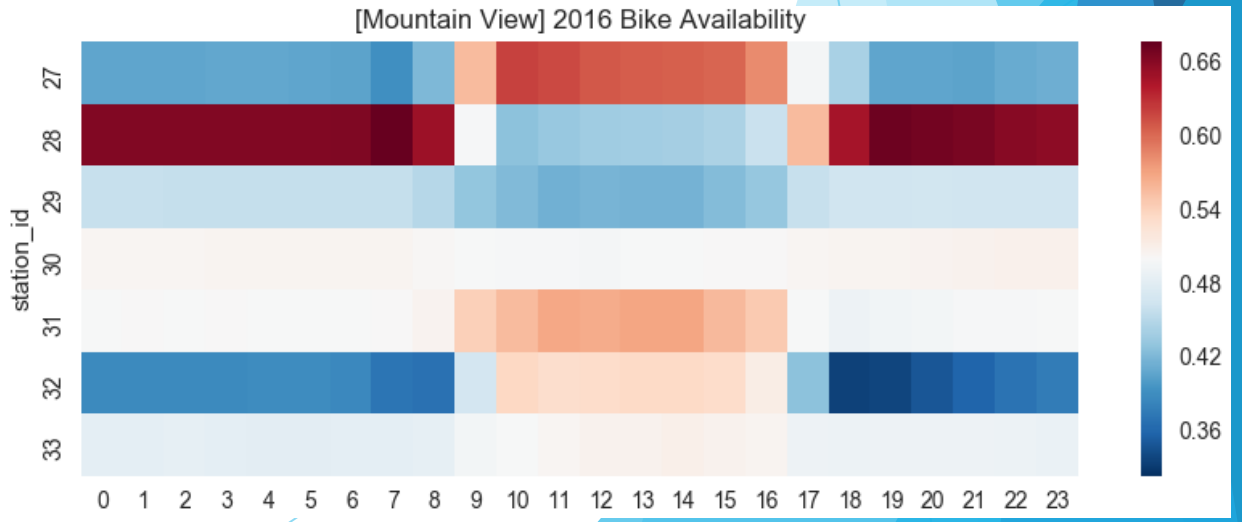
Source code: https://github.com/hidecha/Bike/blob/master/stauts_upd.py
https://github.com/hidecha/Bike/blob/master/status_zero.py

Statusデータを元にした現状分析2

- 次に時間帯ごとの傾向を見るためStatusデータをステーションごとに “bike_availability” を算出

$$\text{bike\_availability} = \frac{\text{bikes\_available}}{\text{bikes\_available} + \text{docks\_available}}$$

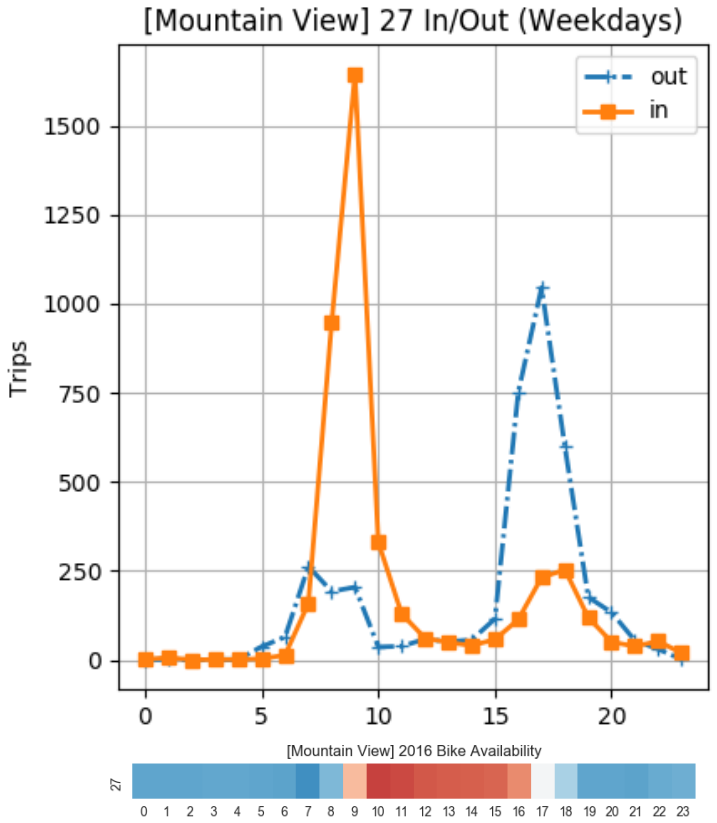
  - 0 に近いほど自転車が不足
  - 1 に近いほどドックが不足 (自転車が過剰)
- 右下図は 2016年平日 Mountain View における bike_availability の平均値を時間ごとに集計したもの
- 朝→昼→晩における自転車数の増減は、ステーションごとに異なることが分かる
  - 27: 少→多→少
  - 28: 多→やや少→多
  - 30: ほぼ変化なし
  - 32: 少→やや多→かなり少、など



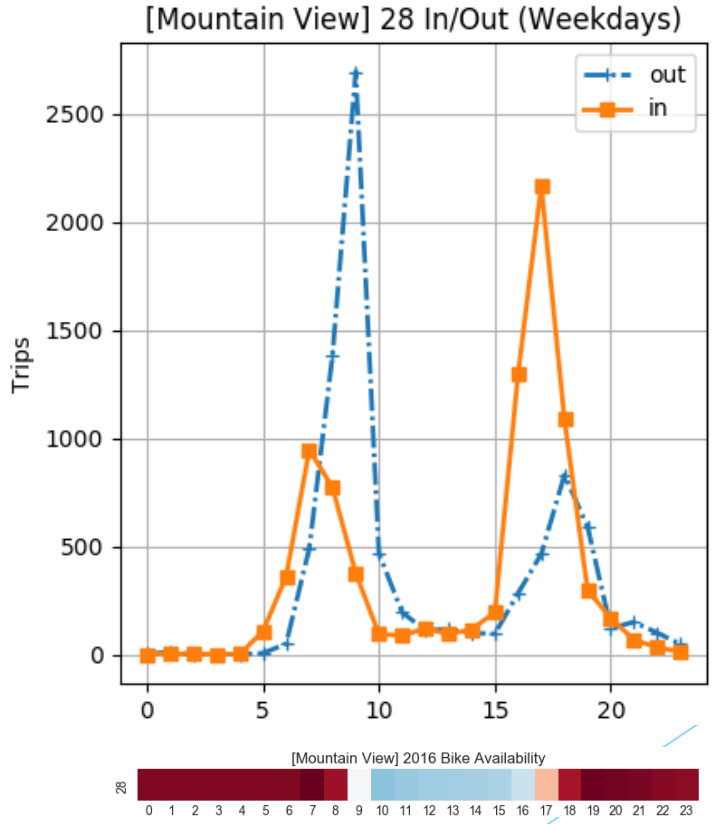
Source code: https://github.com/hidecha/Bike/blob/master/status_heatmap.py

Trip / Status データ比較による考察例

- ステーション ID: 27
  - 朝8時までは自転車が不足気味
  - 8~9時で In >> Out が発生し、昼間は自転車が過剰 (ドックが不足) 気味
  - 17時~18時では逆に In << Out が発生し、再度自転車が不足気味



- ステーション ID: 28
  - 朝8時までは自転車が過剰 (ドックが不足) 気味
  - 8~9時で In << Out が発生し、昼間は自転車が不足気味
  - 17時~18時では逆に In >> Out が発生し、再度自転車が過剰気味



想定される現状の問題点と解決案

bike_availability の偏りによるユーザーへの影響

- 0に近い時間帯が長いほど「自転車に乗りたくても乗れない」というユーザーが増加
- 1に近い時間帯が長いほど「自転車を止めたくても止められない」というユーザーが増加
- なるべくbike_availability=0.5に近づくように再配置 (rebalancing) が必要

現実的な解決案

- 運営側のトラックで自転車をまとめて移動するには人的コスト・輸送コストがかかり、コストを抑えるとなると再配置可能な範囲が限られてくる
- **ユーザーに再配置を手伝ってもらうことはできないか？**
  - ステーションごとのbike_availability の偏りを機械学習により予測し、再配置の貢献度を定義する。
  - 供給不足を補うよう自転車の移動に貢献してくれたユーザーへ報酬を与えることにより偏りを軽減できるのではないか？

データ前処理

説明変数

- Status データに次のパラメーターを追加し、説明変数とする
  - time を year(年), isoweek (年初より第何週), weekday (曜日), hour(時), min(分) に分割
  - isoweek, weekdayを使用した理由は年ごとのカレンダーによる曜日の違いを吸収し、月よりもデータの幅を縮め、精度を上げるのが狙い
  - ただし単純化のため祝日は考慮しない
- 天候による影響も考慮するため、日付と都市をキーとして weather データを結合する
  - Sunny: 晴れの日かどうか、Events = Nanの場合1、それ以外 0
  - Mean_Temperature_F: 平均気温、華氏から摂氏に変換
  - Mean_Wind_Speed_MPH: 平均風速

目的変数

- bike_availability = bikes_available / (bikes_available + docks_available)

Source code: https://github.com/hidecha/Bike/blob/master/weather_upd.py
https://github.com/hidecha/Bike/blob/master/stauts_upd.py

ステーションごとの需要予測モデル1

- すべての Statusデータを結合し、train / test データに分割
  - test_size=0.1, random_state=0 とする
- 今まで学んだモデルをそれぞれ使用し、ステーションごとにスコアを算出
  - ロジスティクス回帰: LogisticRegression
  - 決定木: DecisionTreeClassifier
  - k-NN: KNeighborsClassifier (n_neighbors=6)
  - ランダムフォレスト: RandomForestClassifier
- それぞれのモデルにおけるパラメーターは既定値
- クロスバリデーション (5分割) の平均スコアを算出

Source code: https://github.com/hidecha/Bike/blob/master/status_predict_model1_cv.py

ステーションごとの需要予測モデル1 (続)

- bike_availability を目的変数とした需要予測モデルは **過学習に陥りやすい** (下図赤字参照)

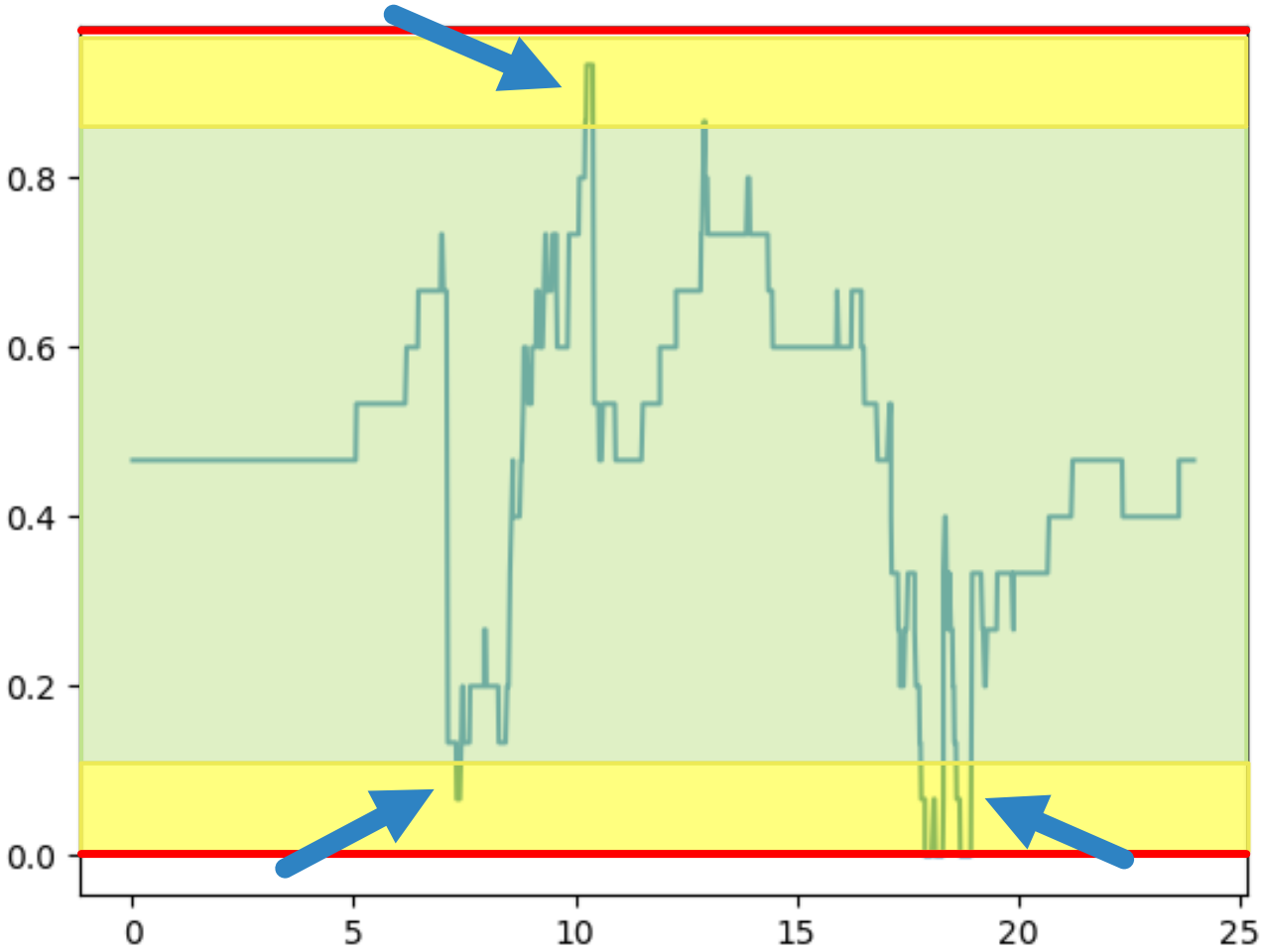
station_id=27
を例とする

Model	Type	Score by bike_availability
LogisticRegression	train	17.2%
	test	17.2%
	cross_val	13.8%
DecisionTreeClassifier	train	99.9%
	test	98.9%
	cross_val	8.8%
KNeighborsClassifier	train	99.0%
	test	97.0%
	cross_val	8.9%
RandomForestClassifier	train	99.9%
	test	98.9%
	cross_val	7.8%

Source code: https://github.com/hidecha/Bike/blob/master/status_predict_model2_cv.py

方針転換

- 回帰問題 → 分類問題



ドックが不足

自転車もドックも足りている

自転車が不足

[San Francisco] 60, 2016-08-31, bike_availability 推移

ステーションごとの需要予測モデル2

- **bike_availability が 0 に近い場合、1に近い場合にフォーカスして予測**
  - bike_availability <= 0.15 の場合、bike_short=1
  - bike_availability >= 0.85 の場合、dock_short=1 と定義
  - それぞれを目的変数とする**分類問題**としての需要予測を行う
- **結果、おおむね 80% 以上の精度が得られる** (下図青字参照)

station_id=27
を例とする

		回帰問題	分類問題
Model	Type	Score by bike_availability	Score by bike_short=1
LogisticRegression	train	17.2%	93.3%
	test	17.2%	93.4%
	cross_val	13.8%	93.3%
DecisionTreeClassifier	train	99.9%	99.8%
	test	98.9%	99.6%
	cross_val	8.8%	80.1%
KNeighborsClassifier	train	99.0%	99.4%
	test	97.0%	99.0%
	cross_val	8.9%	86.7%
RandomForestClassifier	train	99.9%	99.8%
	test	98.9%	99.6%
	cross_val	7.8%	88.0%

Source code: https://github.com/hidecha/Bike/blob/master/status_predict_model2_cv.py

ステーションごとの需要予測モデル2 (続)

- 更に過学習に陥っていないことを確認するため、2016年以外をtrainデータ、2016年をtestデータとして各ステーションごとにスコアを算出
- ベストスコアは概ね 90%以上となり、未知のデータに対しても精度が期待できる
- 集計データ: https://github.com/hidecha/Bike/blob/master/output/DSOC_Supplement.xlsx

都市	予測モデル	Excel タブ
San Jose	クロスバリデーション	[SJ_CrossVal]
	2016年予測	[SJ_2016Pred]
San Francisco	クロスバリデーション	[SF_CrossVal]
	2016年予測	[SF_2016Pred]
Palo Alto	クロスバリデーション	[PA_CrossVal]
	2016年予測	[PA_2016Pred]
Mountain View	クロスバリデーション	[MV_CrossVal]
	2016年予測	[MV_2016Pred]

- 各ステーションごとに最もスコアの良い分析モデルが90%以下のもの: [Pred_Worst]
- これらのステーションは個別にグリッドサーチなどを用いて精度の向上が必要である

Source code: https://github.com/hidecha/Bike/blob/master/status_predict_model2_2016.py

学習データの活用

bike_availability=0.5 に近づくように再配置(rebalancing)するための施策

- bike_short=1 が予測されるステーションおよび時間帯に自転車を drop off したユーザーにポイントを付与
- dock_short=1 が予測されるステーションおよび時間帯に自転車を pick up したユーザーにポイントを付与
- ただし短時間内、同じステーションでの pick up & drop off はノーカウント
- ポイントに応じてバッジ、終日利用券、ギフト券などをプレゼントする
- まずは平日 6~20 時の間で運用して効果測定を行う
- 効果は「ステーションごと bikes_available=0, docks_available=0 となった回数が何%減ったか？」で計測 (p4: Statusデータを元にした現状分析1を参照)

Q&A