

Springboard Data Science Intensive

Instacart Market Basket Competition

Ryan Alexander Alberts
August 1, 2017



Ryan Alexander Alberts

Retail eCommerce & Operations Strategy

Previous

COO, MyToolStore.com (eCommerce retailer)

Market Strategist, Pearce Properties

Education

MIT | Supply Chain Management

Columbia | English Literature

Optimization

JMP, Gurobi,
Excel

Systems Design

ERP & CRM
(NetSuite), SQL

Data Visualization

Tableau, Seaborn,
Matplotlib

Data Science

Python,
APIs, EDIs

Introduction: Retail eCommerce

Disruption

Walmart's acquisition of Jet.com,
Amazon's acquisition of Whole Foods



products experience a wildly different lifecycle
between production and consumption

Strategic Response

Predicting what customers will order next is critical
to managing eCommerce supply chains

Lean, adaptive supply chains require accurate
demand forecasting

Introduction: Kaggle Competitions

Featured Prediction Competition

Instacart Market Basket Analysis

Which products will an Instacart consumer purchase again?

Instacart · 2,623 teams · 21 days ago

\$25,000 Prize Money

Top Platform for Data Science Competitions

Attacks real-world problems using anonymized data from public & private institutions

Extensive public repositories of:

- Machine-learning & Predictive Modelling techniques

- Over 500k Data Scientists
- 2M+ machine-learning models
- Over 200 competitions hosted
- \$10,000 – \$30,000 typical Prize Money
- Collaborative, global user community

Introduction: Instacart

Instacart is a pure-play eCommerce grocer

\$25,000 prize for the top-three demand-forecasting models

The Data:

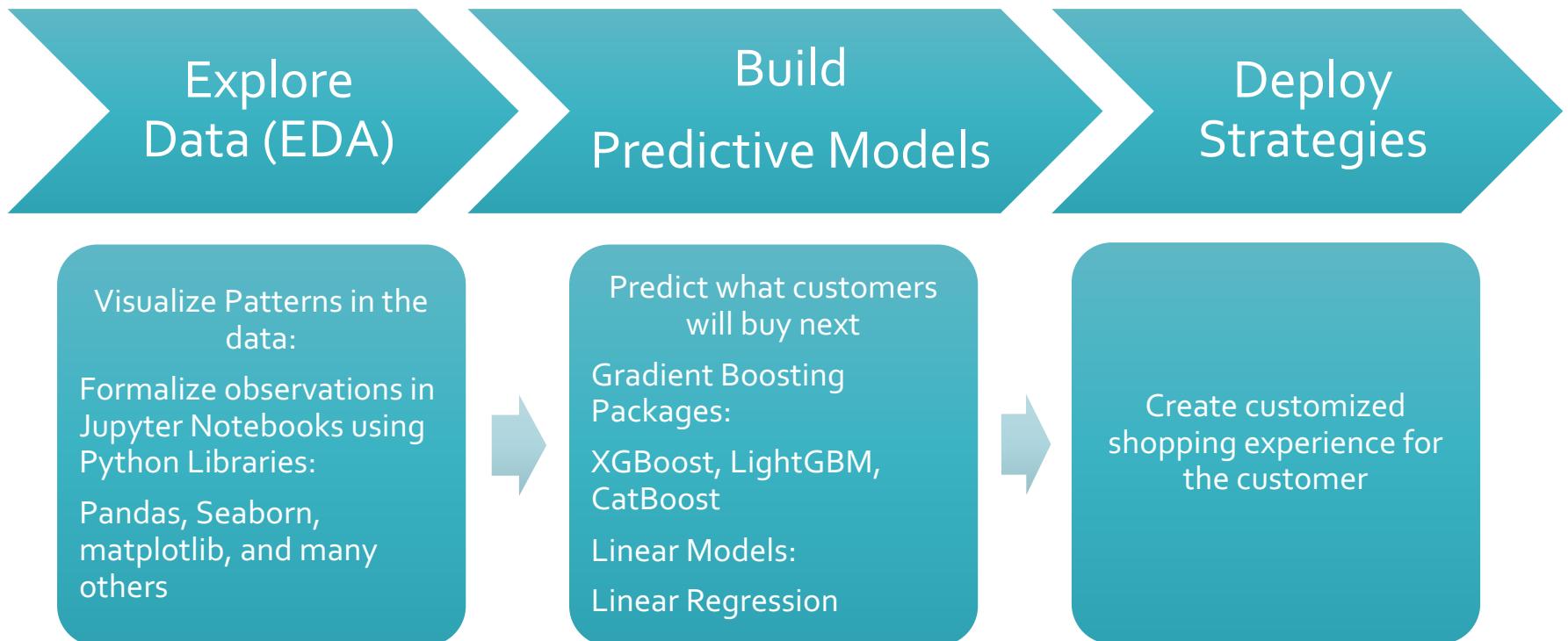
- 3M orders
- 206k customers
- Clean & well-wrangled
- Customer & Order History

Evaluative Metric: Mean F1 Score:

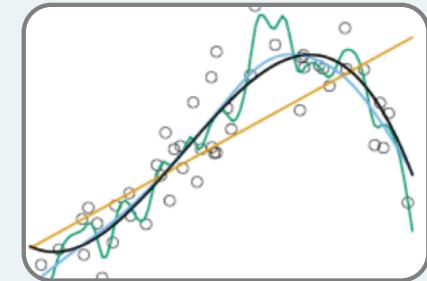
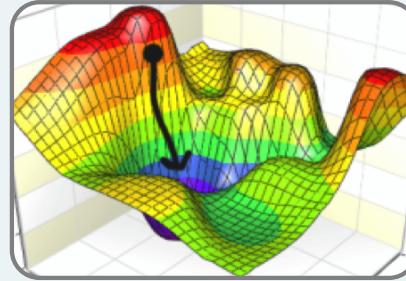
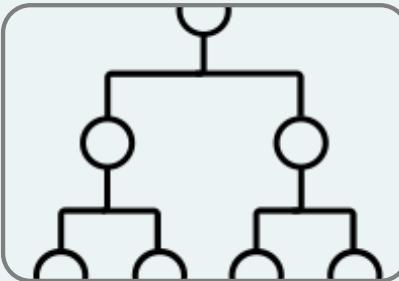
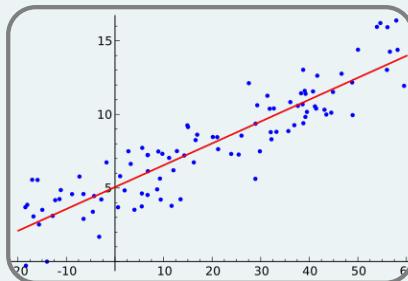
$$\frac{2 * (\text{precision} * \text{recall})}{(\text{precision} + \text{recall})}$$

<https://www.kaggle.com/c/instacart-market-basket-analysis>

Methodology



Models



Logistic Regression

Used to assess feature importance according to its ability to predict binary outcomes

Instacart Market Basket Analysis

Decision Trees

Method of classifying observations by weighing feature importance, closely analogous to how humans actually make decisions

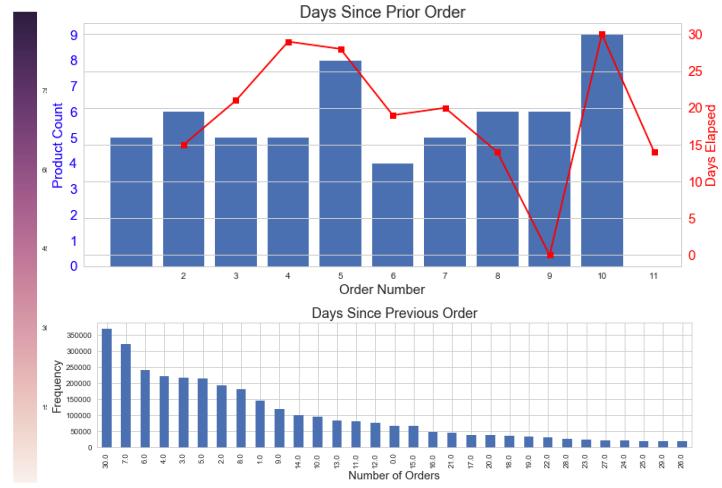
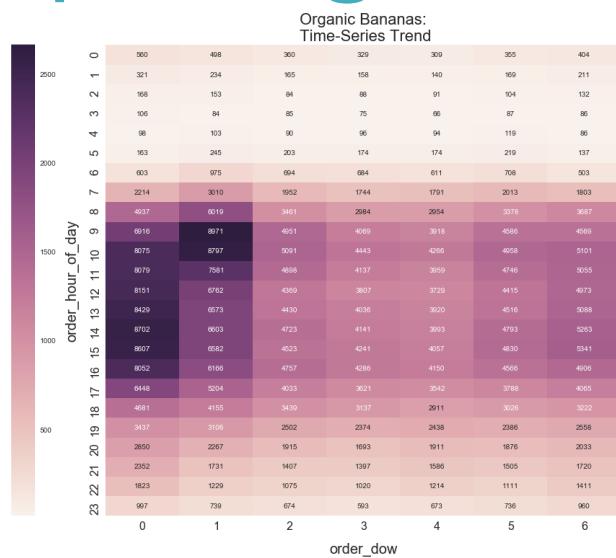
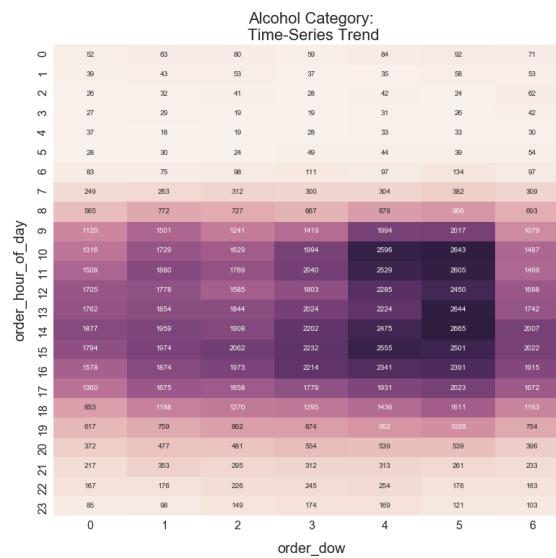
Gradient Boosting

Sequential, dependent parameter analysis using many decision trees to assess feature importance

Ensemble Methods

Combining output of several models. Empirically, most of the winning models in Data Science competitions are ensembles

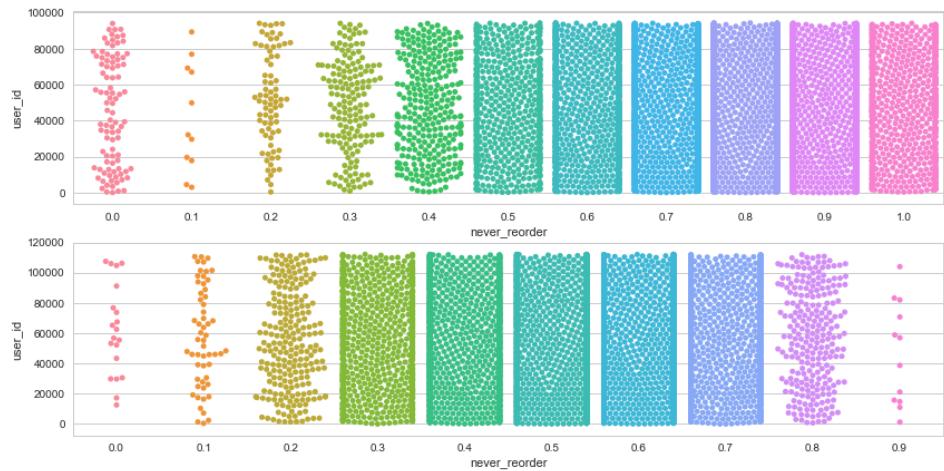
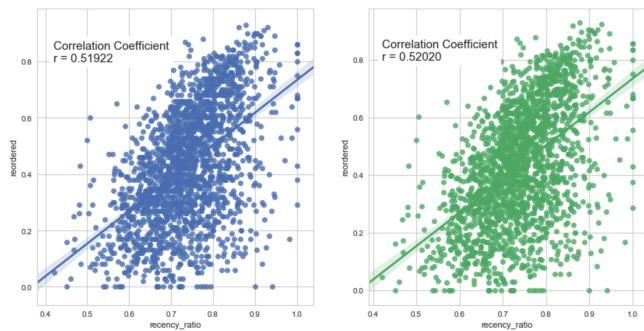
Exploring the Data



Exploring the Data (cont'd.)

```
In [144]: from scipy import stats
fig, ax = plt.subplots(1, 2, figsize=(14,7))
r = stats.pearsonr(t['recency_ratio'][::200], t['reordered'][::200])
r = np.round(r, decimals=2)
bbox_props = dict(fc="w", ec="w", lw=2)
ax[0].text(.42, .8, "Correlation Coefficient\nnr = 0.51922",
           size=15,
           bbox=bbox_props)
ax[1].text(.42, .8, "Correlation Coefficient\nnr = 0.52020",
           size=15,
           bbox=bbox_props)

t = pd.DataFrame(recent_four.groupby('user_id').mean()).reset_index()
u = t.copy()
t['reordered'] = np.round(t['reordered'], decimals=2)
sns.regplot(x="recency_ratio", y="reordered", data=t[:2000], ax=ax[0]);
sns.regplot(x="recency_ratio", y="reordered", data=u[:2000], ax=ax[1]);
```



Finding Patterns in the Data

```
In [20]: # TEST SET

print 'Processing recency features --Approx. 40min -- ...'
sequence = pd.DataFrame(TEST__user_orders.groupby(['user_id',
                                                    'product_id',
                                                    'order_number'])
                        .size())
sequence.reset_index()

def TEST__user_orders(sequence):
    sequence = sequence.drop([0], axis=1)

    basket_by_order = pd.DataFrame(sequence.groupby(['user_id',
                                                    'order_number'])
                                    .size())
    basket_by_order.reset_index()
    basket_by_order.rename(columns = {0 : 'order_size'}, inplace=True)
    sequence.merge(basket_by_order,
                   on=['user_id',
                       'order_number'],
                   how='left')

    recent_four = pd.DataFrame(sequence.groupby(['user_id', 'product_id'])
                                .size())
    recent_four['order_number'].apply(lambda x:
                                       x.nlargest(4).values))

print 'Processing frequency features ...'

In [97]: #m = pd.DataFrame(TEST__user_orders.groupby('user_id')['order_number'].max()).reset_index()
#m.rename(columns = {'order_number' : 'prior_orders'}, inplace=True)
recent_four = recent_four.merge(m, on='user_id', how='left')
recent_four['recency_ratio'] = (recent_four['most_recent'] / recent_four['prior_orders']).astype(np.float32)
recent_four = recent_four.merge(TEST__user_orders[['user_id',
                                                    'product_id',
                                                    'reordered']], on=['user_id',
                                                               'product_id'], how='left')

recent_four
```

id	product_id	order_number	most_recent	second_most_recent	third_most_recent	fourth_most_recent	never_reordered	avg_no_reordered	recency_ratio	reordered
3	248	[2]	2	-1	-1	-1	1	0.424242	0.166667	0
3	1005	[10]	10	-1	-1	-1	1	0.424242	0.833333	0
3	1819	[7, 6, 4]	7	6	4	-1	0	0.424242	0.583333	0
3	1819	[7, 6, 4]	7	6	4	-1	0	0.424242	0.583333	1
3	1819	[7, 6, 4]	7	6	4	-1	0	0.424242	0.583333	1
3	7503	[3]	3	-1	-1	-1	1	0.424242	0.250000	0
3	8021	[2]	2	-1	-1	-1	1	0.424242	0.166667	0
3	9387	[7, 6, 5, 4]	7	6	5	4	0	0.424242	0.583333	0
3	9387	[7, 6, 5, 4]	7	6	5	4	0	0.424242	0.583333	1
3	9387	[7, 6, 5, 4]	7	6	5	4	0	0.424242	0.583333	1
3	9387	[7, 6, 5, 4]	7	6	5	4	0	0.424242	0.583333	1
3	9387	[7, 6, 5, 4]	7	6	5	4	0	0.424242	0.583333	1
3	12845	[4]	4	-1	-1	-1	1	0.424242	0.333333	0
3	14992	[7, 6]	7	6	-1	-1	0	0.424242	0.583333	0
3	14992	[7, 6]	7	6	-1	-1	0	0.424242	0.583333	1
3	15143	[1]	1	-1	-1	-1	1	0.424242	0.083333	0
3	16207	[0, 7, -1]	0	-1	-1	-1	0	0.424242	0.750000	0

Example: Recent order behavior matters more

Use domain knowledge to isolate meaningful patterns

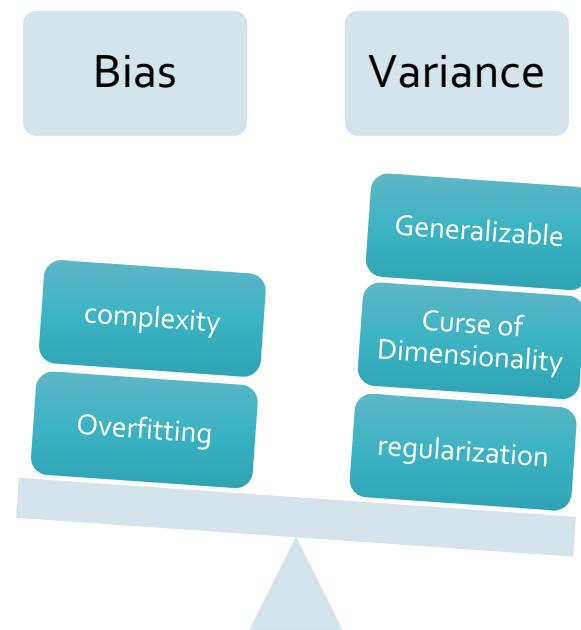


Engineer a feature that captures the pattern



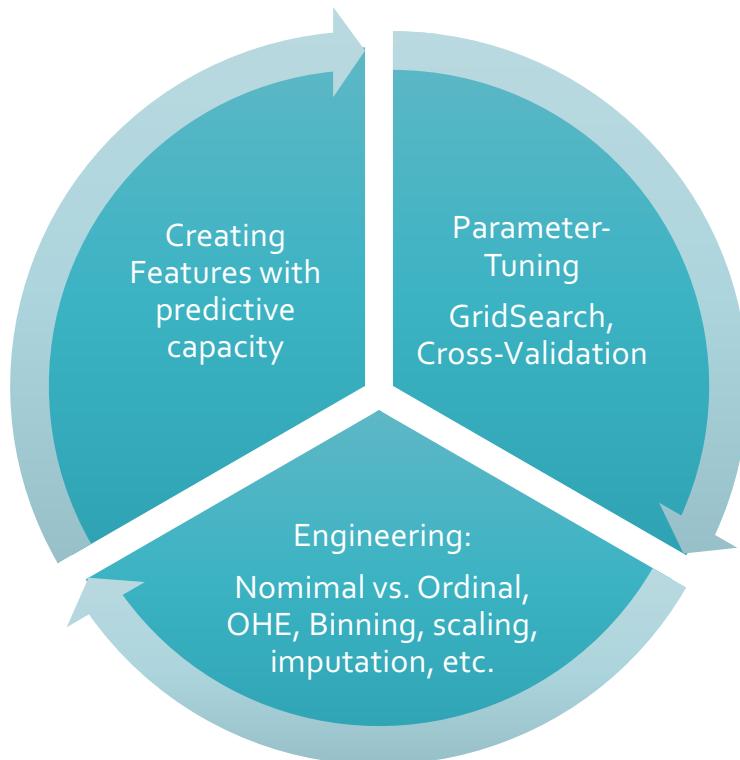
Build feature set and feed into model

Feature Engineering & Parameter-Tuning

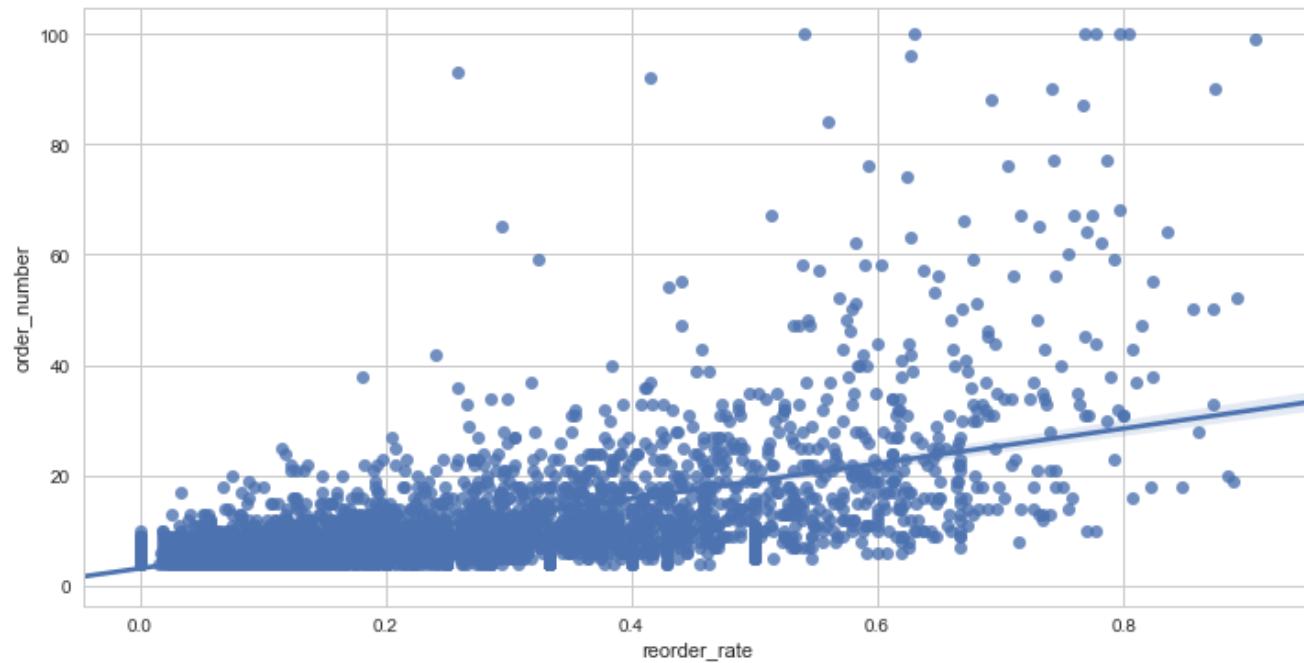


(Mean Squared) Error = Bias + Variance
“All models are wrong, some models are useful”

Feature Engineering & Parameter-Tuning



When No Prediction is the Best Prediction



Optimizing F-Measures for Binary Classifier

```
try:
    test_df = test_df.drop(['usr_threshold'], axis=1)
except:
    pass

class F1Optimizer():
    def __init__(self):
        pass

    @staticmethod
    def get_expectations(P, pNone=None):
        expectations = []
        P = np.sort(P)[::-1]

        n = np.array(P).shape[0]
        DP_C = np.zeros((n + 2, n + 1))
        if pNone is None:
            pNone = (1.0 - P).prod()

        DP_C[0][0] = 1.0
        for j in range(1, n):
            DP_C[0][j] = (1.0 - P[j - 1]) * DP_C[0, j - 1]

        for i in range(1, n + 1):
            DP_C[i, i] = DP_C[i - 1, i - 1] * P[i - 1]
            for j in range(i + 1, n + 1):
                DP_C[i, j] = P[j - 1] * DP_C[i - 1, j - 1] + (1.0 - P[j - 1]) * DP_C[i, j - 1]

        DP_S = np.zeros((2 * n + 1,))
        DP_SNone = np.zeros((2 * n + 1,))
        for i in range(1, 2 * n + 1):
            DP_S[i] = 1. / (1. * i)
            DP_SNone[i] = 1. / (1. * i + 1)
        for k in range(n + 1)[::-1]:
            f1 = 0
            f1None = 0
            for k1 in range(n + 1):
                f1 += 2 * k1 * DP_C[k1][k] * DP_S[k + k1]
                f1None += 2 * k1 * DP_C[k1][k] * DP_SNone[k + k1]
            for i in range(1, 2 * k - 1):
                DP_S[i] = (1 - P[i - 1]) * DP_S[i] + P[k - 1] * DP_S[i + 1]
                DP_SNone[i] = (1 - P[k - 1]) * DP_SNone[i] + P[k - 1] * DP_SNone[i + 1]
            expectations.append([f1None + 2 * pNone / (2 * k), f1])

        #print '\nexpectations', np.array(expectations[::-1]).T

        return np.array(expectations[::-1]).T

    @staticmethod
    def maximize_expectation(P, pNone=None):
        expectations = F1Optimizer.get_expectations(P, pNone)
        ix_max = np.unravel_index(expectations.argmax(), expectations.shape)
        max_f1 = expectations[ix_max]
        predNone = True if ix_max[0] == 0 else False
        best_k = ix_max[1]
```

Basket Size: Predicting HOW MANY is as important as WHICH products a customer will buy next

Problem: How do you rank the predictions, and what threshold do you use?

Solution: Maximize the expected F1 Score and rank product matrix

Conclusions & Recommendations

Use Predictions from the model to:

- Refine search recommendations for users while they are shopping
- More effectively manage inventory
- Create targeted merchandizing campaigns
- More effectively engage customers with personalized content

Thank you!

A screenshot of a web browser displaying a Kaggle profile and a calendar. The profile on the left shows a photo of a man, his name 'Ryan A.', and some basic information: 'student at MIT' from 'New York, New York, United States'. He joined 2 months ago and was last seen in the 'Competitions' section. Below the profile are links for 'Home', 'Competitions (1)', 'Kernels (2)', 'Discussion (1)', and 'Datasets'. The right side features a monthly calendar for August 2017. The days of the week are labeled at the top: Sun, Mon, Tue, Wed, Thu, Fri, Sat. The dates range from 30 (Sun) to 9 (Sat). A red dot highlights the date 'Aug 8'. The date 'Aug 1' is also highlighted with a red dot. The date 'Aug 17' has a small red icon next to it. The date 'Sep 1' is also highlighted with a red dot. The date 'Aug 4' is labeled 'Labor Day'. At the bottom of the browser window, there is a code editor with the following text:

```
In [ ]: #min-max scaling -- potential candidates...
```

...Final Rank*

The screenshot shows a Kaggle user profile for 'Ryan A.'. The profile header includes a photo of a man in a suit, the name 'Ryan A.', and a brief bio: 'student at MIT, New York, New York, United States'. It also shows 'Joined 3 months ago · last seen in the past day' and social links for LinkedIn and GitHub. The 'Competitions Contributor' badge is prominently displayed. Below the header, there's a navigation bar with 'Home' (underlined) and other tabs: 'Competitions (1)', 'Kernels (2)', 'Discussion (1)', 'Datasets (0)', and '...'. An 'Edit Profile' button is on the right. The main content area is divided into three columns: 'Competitions Contributor' (Unranked, 0 points), 'Kernels Contributor' (Unranked, 0 points), and 'Discussion Contributor' (Unranked, 0 points). The 'Competitions' column contains a link to an Instacart Market Basket Analysis competition, which Ryan placed 479th out of 2623 participants.

* Competition ended after the start of my Master's program, so I made the decision to stop work on the project August 8th.