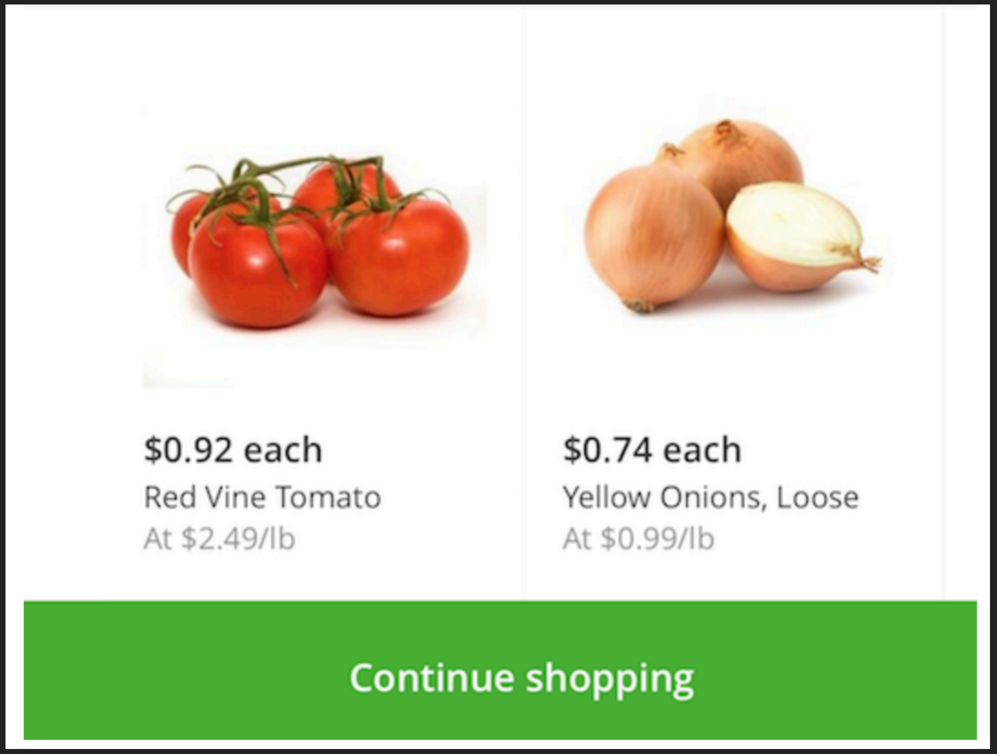
PREDICTING CUSTOMER ORDERS WITH INSTACART

WHY AM I DOING THIS?

Vladimir in a cart

Instacart



- Business problem: predict which products will be in a user's next order.
- Application: optimize supply chains and reduce waste.

DATA AND TOOLS

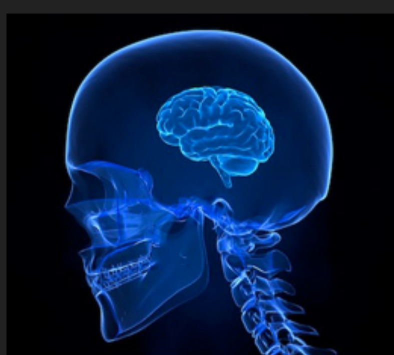
- Instacart Kaggle competition data https://www.kaggle.com/c/instacart-market-basket-analysis/data
- Python libraries, such as sklearn, xgboost, seaborn, etc.
- Amazon Web Services for training models and testing
- Tableau for visualization
- Structured Query Language for data management

PRELIMINARY MODEL SELECTION

- KNN:
  - few features, users can be “clustered”
- Logistic Regression:
  - pocket pick for binary classification, interpretable
- Naive Bayes
  - features fairly independent, some are textual
- XGBoost because Joe likes it

AMERICA'S NEXT TOP (MACHINE LEARNING) MODEL

Logistic Regression: good results, improved with features, handled class imbalance, added complexity



Joe's notebook

Engineer more features

Scale features, add complexity

Select best class weights

Train, validate, test

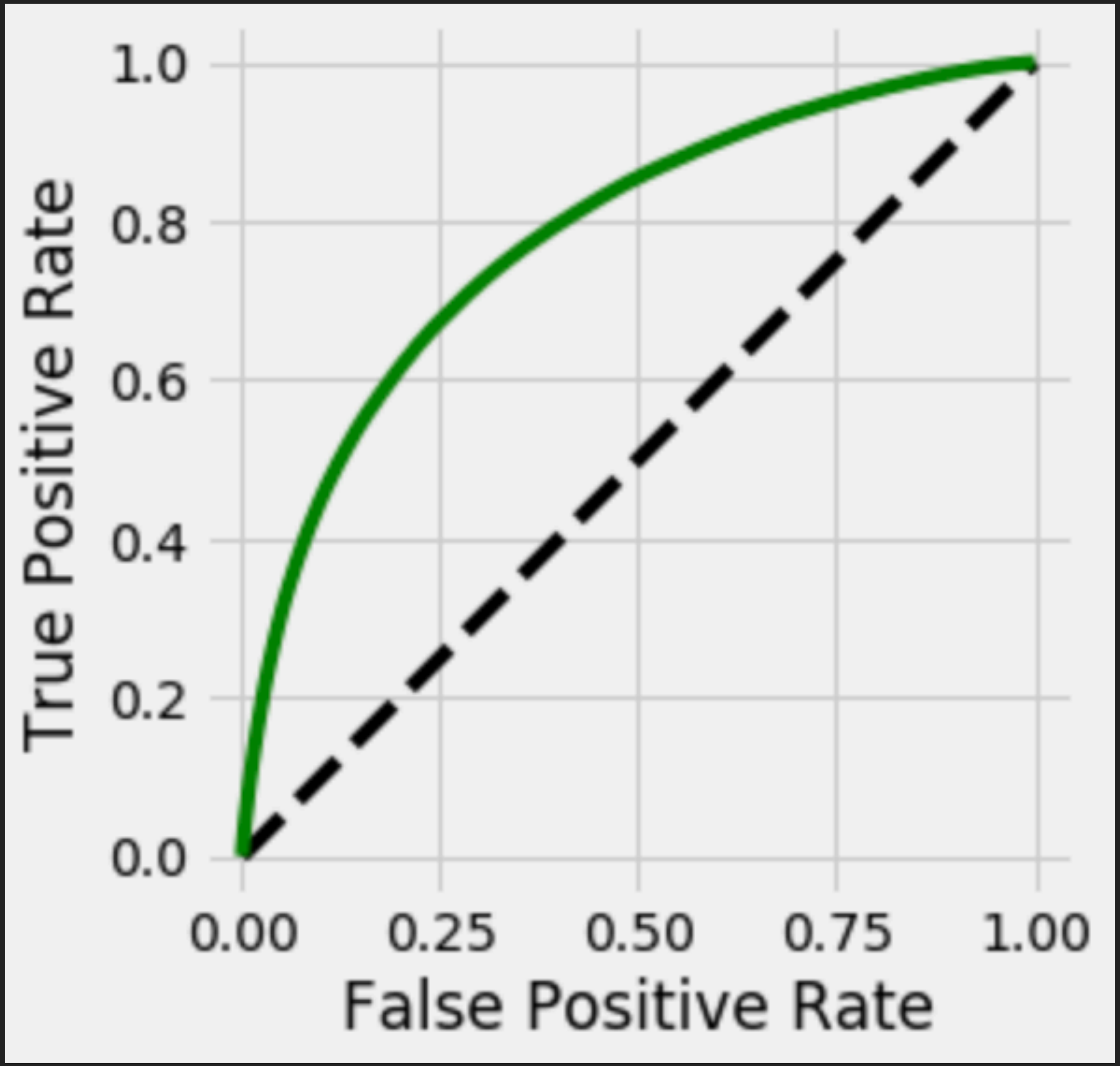
THE ULTIMATE FORM!

- Simple and interpretable!
- 13 new features!
- Custom train/test splitting
- All features scaled to [0,1] interval
- Hand-picked class weights {1 : 6, 0 : 1}!
- F1: 0.3795 on holdout validation data
- F1: 0.3812 on holdout test data 😒

MORE METRICS

- Training Accuracy: 0.84
- Test Accuracy: 0.83
- Predicting reorder:
  - Precision: 0.30
  - Recall: 0.52
- Predicting no reorder:
  - Precision: 0.94
  - Recall: 0.87

ROC curve



COEFFICIENT INTERPRETATION

Change in the odds the product will be reordered with each unit increase in the feature.

- Frequency: 25%
- User total orders: 16%
- Product total orders: 8%
- Add to cart order: -10%
- User vs average deviation: -7%

Coefficients, exponentiated

user_product_total_orders	1.156720
product_total_orders	1.075510
product_avg_add_to_cart_order	0.905234
user_total_orders	0.981141
user_avg_cartsize	1.051901
user_total_products	0.994552
user_avg_days_since_prior_order	0.986311
user_product_avg_add_to_cart_order	0.946543
user_product_order_freq	1.250403
product_avg_order_dow	0.965113
product_avg_order_hour_of_day	1.030801
product_avg_days_since_prior_order	0.994743
user_avg_order_dow	0.991134
user_avg_order_hour_of_day	1.013924
user_product_avg_days_since_prior_order	1.004527
user_product_avg_order_dow	0.979908
user_product_avg_order_hour_of_day	1.011983
product_total_orders_delta_per_user	0.929793
product_avg_add_to_cart_order_delta_per_user	0.956359
product_avg_order_dow_per_user	0.984902
product_avg_order_hour_of_day_per_user	1.018596
product_avg_days_since_prior_order_per_user	0.990260

CONCLUSION

- Pros:
  - Easy to interpret, predict
  - Not too computation heavy
  - Handles class imbalance
  - Low variance
- Cons:
  - Low predictive power
  - Low F1 score (0.3928)
  - Low precision (0.3)
  - (high bias) 😭

Low F1 score value means the company will be losing money and wasting products, which is unethical. 💁🏼‍♂️