Addressing Barclays Center's Unsold Tickets to Nets NBA Games

By: Horace Fung, Dylan Prada, Nicole Rodriguez, Amber Wang

Business Understanding

The Barclays Center must host every home game for the NBA Basketball team Brooklyn Nets, and incur operating cost regardless of how many fans attend the game. Hence, the business is highly sensitive to ticket sales, and wants to know when a ticket would be unsold in order to take action and fill the seat. The business problem is whether we can predict unsold tickets through a supervised classification model, and generate profits by promoting the ticket/bundle through spending extra marketing resources. To be more specific, our target variable is the probability that a Nets home-game ticket/bundle will be unsold at any given time within 1 week period before the game day. Our data's base rate for unsold tickets is 9.48%, which means the data is heavily skewed and conventional measures like accuracy may be a bad measure of performance (more on that later). We believe that we can leverage the data mining process to design a solution that lowers unsold tickets with respect to Barclays Center's goal, maximizing profit at the lowest cost. We envision this to be a low budget and relatively easy to implement solution.

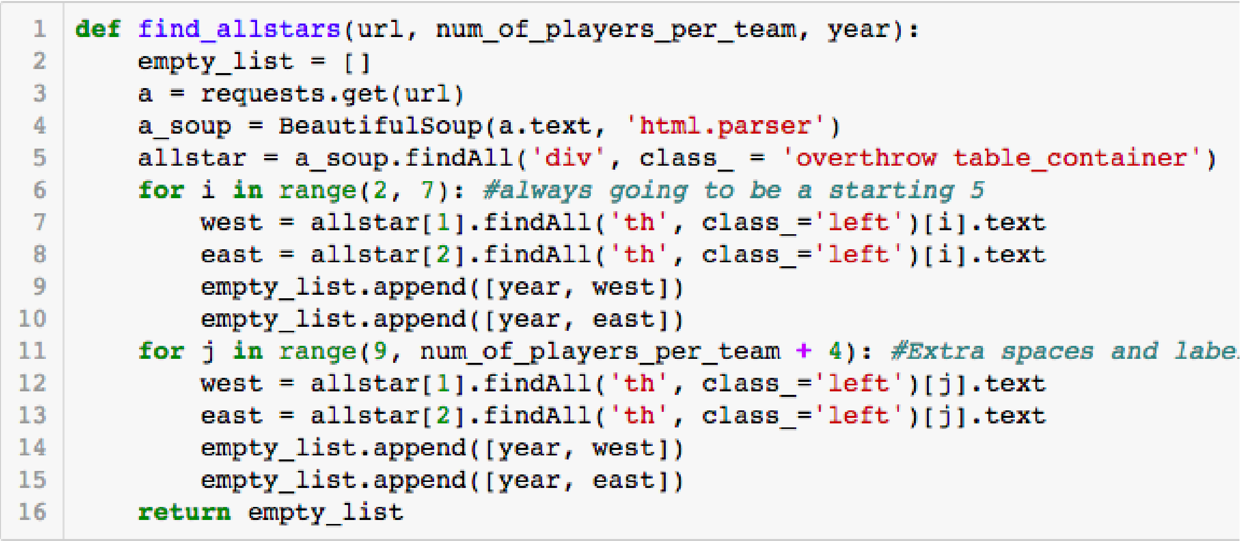
Data Understanding + Data Preparation

Notebooks: *Scrapping BasketballReference.com.ipynb, Combining_Data.ipynb, Cleaning_Data.ipynb*

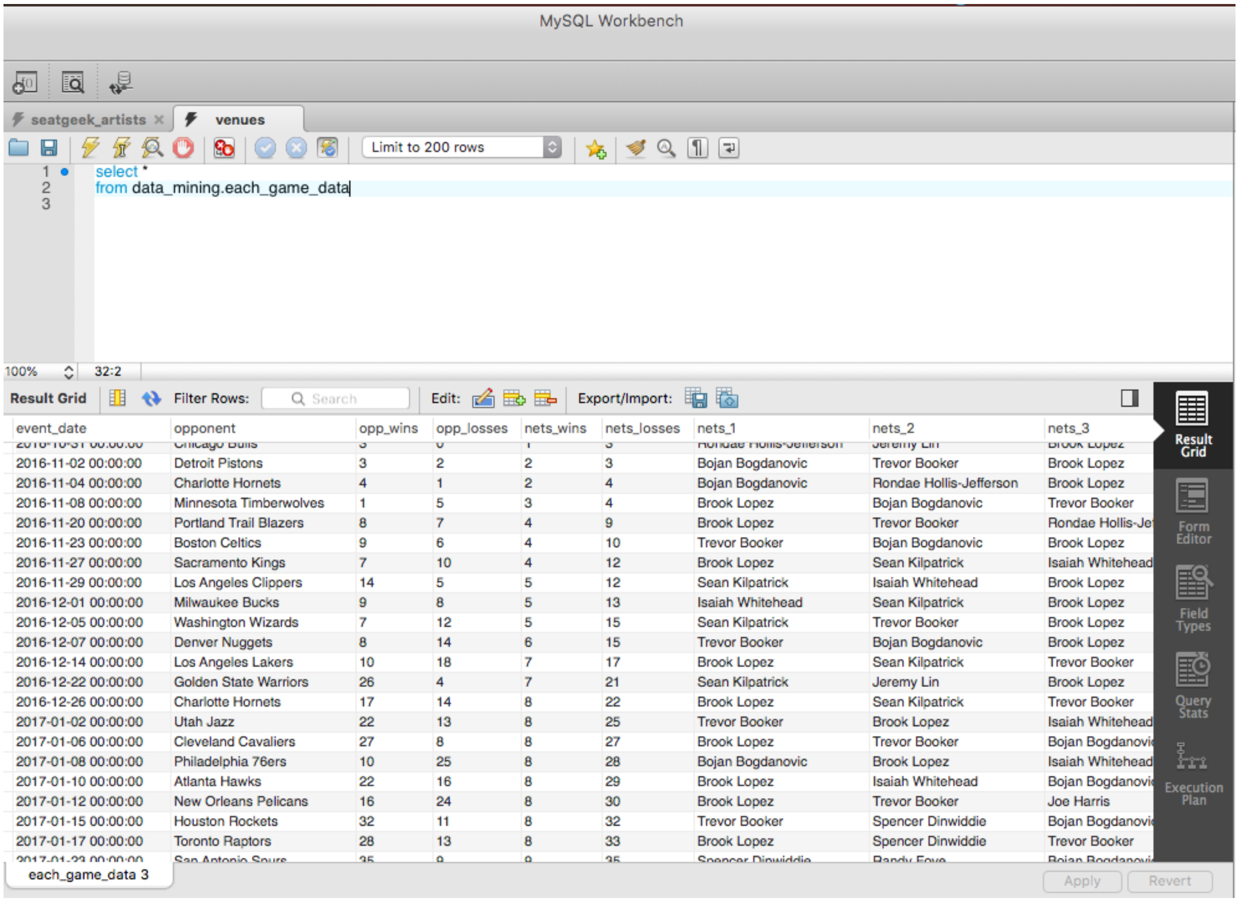
Overall, we wanted data that can capture two elements that we believe drive tickets sale. First, information about the event and venue like seat location or day of the week. Second, information about the basketball context such as the popularity of the away team.

The centerpiece in our data set is information on Barclays Center's ticket sales for the Nets' 2017-2016 season. This dataset contains labels for *(Sold/Not Sold)* tickets. Other variables include *(event_date, opponent, section, row, beg_seat, end_seat, price_per_ticket, bundle_price)*. The seat location variables reflect different quality of the tickets, as some seats like courtside are usually preferred. In terms of the basketball context, we looked at possible drivers of ticket sales such as the presence of star players, championship contending teams, current team performances and recent success from either side. The Nets have been struggling in the past few years, and it is likely that some ticket holders will go just to see certain visiting teams and players. Therefore, we scrapped BasketballReference.com for additional data. The scraped data can be separated into four sections— 1) Data on the game that corresponds to the ticket, 2) Past conference standings, 3) Past playoff performance and 4) All-star players. The reasoning behind the all-star section is that we think performance and popularity of teams are highly correlated— but we need to account for certain situations such as when a popular player is on a poorly performing team (Carmelo Anthony, Knicks 2016), in which case they can still attract ticket sales. The all-star attribute can help incorporate this relationship. Below is a sample code of how we collected the data, and how we store:

Function to get a list of all-stars



```python
def find_allstars(url, num_of_players_per_team, year):
    empty_list = []
    a = requests.get(url)
    a_soup = BeautifulSoup(a.text, 'html.parser')
    allstar = a_soup.findAll('div', class_ = 'overthrow table_container')
    for i in range(2, 7): #always going to be a starting 5
        west = allstar[1].findAll('th', class_='left')[i].text
        east = allstar[2].findAll('th', class_='left')[i].text
        empty_list.append([year, west])
        empty_list.append([year, east])
    for j in range(9, num_of_players_per_team + 4): #Extra spaces and labe
        west = allstar[1].findAll('th', class_='left')[j].text
        east = allstar[2].findAll('th', class_='left')[j].text
        empty_list.append([year, west])
        empty_list.append([year, east])
    return empty_list
```



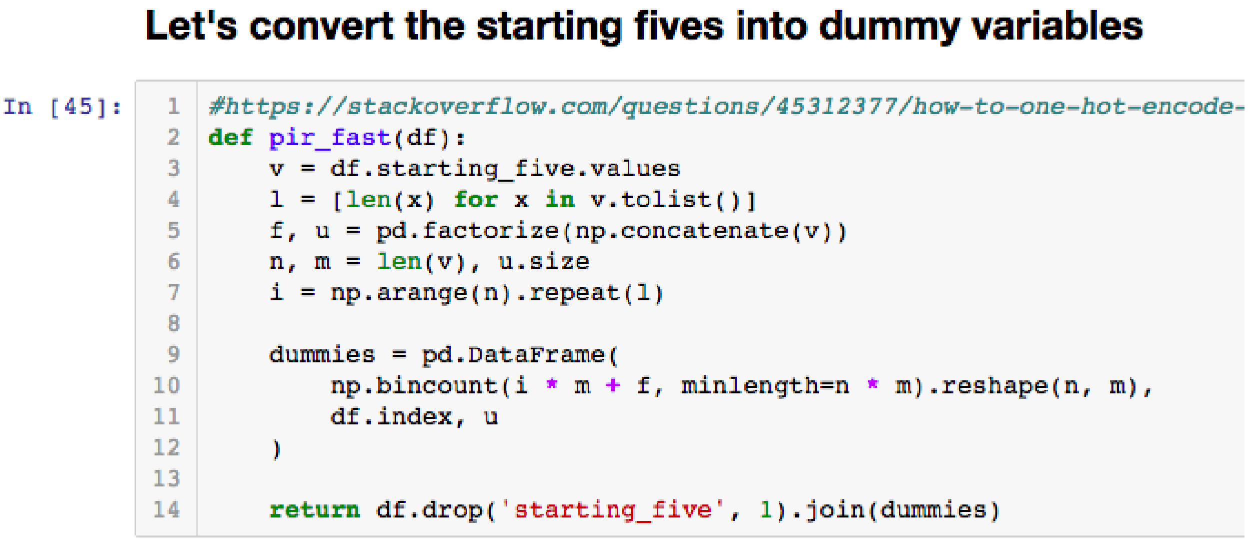
```sql
select *
from data_mining.each_game_data
```

event_date	opponent	opp_wins	opp_losses	nets_wins	nets_losses	nets_1	nets_2	nets_3
2016-10-31 00:00:00	Chicago Bulls	3	0	1	3	Rondae Hollis-Jefferson	Jeremy Lin	Brook Lopez
2016-11-02 00:00:00	Detroit Pistons	3	2	2	3	Bojan Bogdanovic	Trevor Booker	Brook Lopez
2016-11-04 00:00:00	Charlotte Hornets	4	1	2	4	Bojan Bogdanovic	Rondae Hollis-Jefferson	Brook Lopez
2016-11-08 00:00:00	Minnesota Timberwolves	1	5	3	4	Brook Lopez	Bojan Bogdanovic	Trevor Booker
2016-11-20 00:00:00	Portland Trail Blazers	8	7	4	9	Brook Lopez	Trevor Booker	Rondae Hollis-Je
2016-11-23 00:00:00	Boston Celtics	9	6	4	10	Trevor Booker	Bojan Bogdanovic	Brook Lopez
2016-11-27 00:00:00	Sacramento Kings	7	10	4	12	Brook Lopez	Sean Kilpatrick	Isaiah Whitehead
2016-11-29 00:00:00	Los Angeles Clippers	14	5	5	12	Sean Kilpatrick	Isaiah Whitehead	Brook Lopez
2016-12-01 00:00:00	Milwaukee Bucks	9	8	5	13	Isaiah Whitehead	Sean Kilpatrick	Brook Lopez
2016-12-05 00:00:00	Washington Wizards	7	12	5	15	Sean Kilpatrick	Trevor Booker	Brook Lopez
2016-12-07 00:00:00	Denver Nuggets	8	14	6	15	Trevor Booker	Bojan Bogdanovic	Brook Lopez
2016-12-14 00:00:00	Los Angeles Lakers	10	18	7	17	Brook Lopez	Sean Kilpatrick	Trevor Booker
2016-12-22 00:00:00	Golden State Warriors	26	4	7	21	Sean Kilpatrick	Jeremy Lin	Brook Lopez
2016-12-26 00:00:00	Charlotte Hornets	17	14	8	22	Brook Lopez	Sean Kilpatrick	Trevor Booker
2017-01-02 00:00:00	Utah Jazz	22	13	8	25	Trevor Booker	Brook Lopez	Isaiah Whitehead
2017-01-06 00:00:00	Cleveland Cavaliers	27	8	8	27	Brook Lopez	Trevor Booker	Bojan Bogdanovi
2017-01-08 00:00:00	Philadelphia 76ers	10	25	8	28	Bojan Bogdanovic	Brook Lopez	Isaiah Whitehead
2017-01-10 00:00:00	Atlanta Hawks	22	16	8	29	Brook Lopez	Isaiah Whitehead	Bojan Bogdanovi
2017-01-12 00:00:00	New Orleans Pelicans	16	24	8	30	Brook Lopez	Trevor Booker	Joe Harris
2017-01-15 00:00:00	Houston Rockets	32	11	8	32	Trevor Booker	Spencer Dinwiddie	Bojan Bogdanovi
2017-01-17 00:00:00	Toronto Raptors	28	13	8	33	Brook Lopez	Spencer Dinwiddie	Trevor Booker
2017-01-23 00:00:00	San Antonio Spurs	35	9	9	35	Spencer Dinwiddie	Randy Foye	Bojan Bogdanovi

The final data set has 58,418 instances and 23 features. The features are:

Begseat, *endseat*, *quantity*, *year*, *month*, *dayofweek*, *time_left*, *price*, *nets_losses*, *nets_wins*, *opp_losses*, *opp_wins*, *allstars_count*, *rounds*, *oppg*, *ppg*, *seed*, *winloss*, *nets_wins*, *nets_losses*, *starting_five*, *section* and *row*.

Categorical features like *starting_five* or *section* are encoded as dummy variables.



Let's convert the starting fives into dummy variables

In [45]:

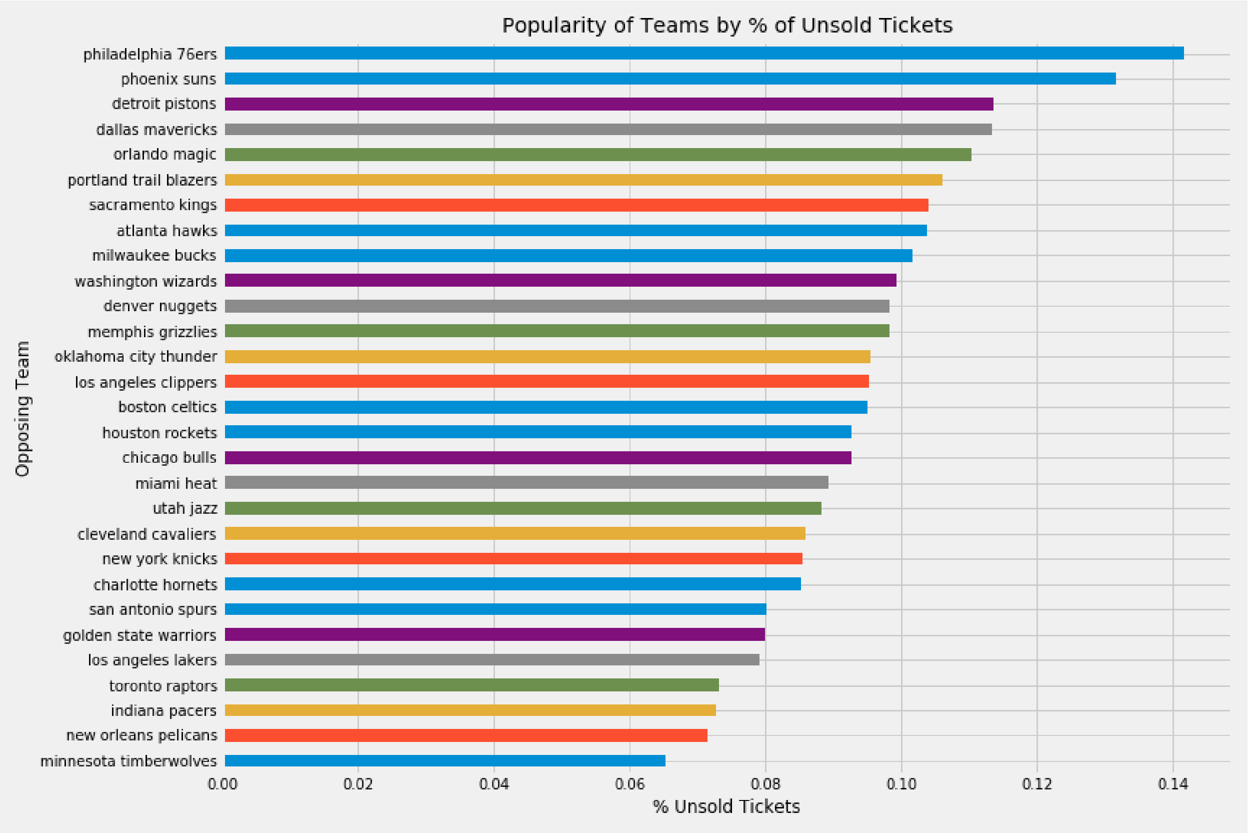
```
#https://stackoverflow.com/questions/45312377/how-to-one-hot-encode-
def pir_fast(df):
    v = df.starting_five.values
    l = [len(x) for x in v.tolist()]
    f, u = pd.factorize(np.concatenate(v))
    n, m = len(v), u.size
    i = np.arange(n).repeat(l)

    dummies = pd.DataFrame(
        np.bincount(i * m + f, minlength=n * m).reshape(n, m),
        df.index, u
    )

    return df.drop('starting_five', 1).join(dummies)
```

Some of the more interesting engineered features include *allstars_count*, which totals the number of allstars in the past 5 years present at the match, *rounds*, which gives an ordinal ranking of the furthest playoff round the opposing team made last year and *time_left* which is how much time was left until game day when a ticket was sold. In general, we were very cautious about leakage. One feature that has problems is *starting_five*. We do not have 100% certainty on who the starting fives are before the game in deployment, unlike our historical data. However, major changes to the starting line up is usually released beforehand, and customers are also basing their decisions to purchase a ticket on expected lineup, not actual lineup. Therefore, we think the up-to-date expected lineup before the game is a suitable proxy. We also found instances of human-error in the Barclays Center dataset. For all of our data, we made sure to create unique indexes if possible— the Barclays Center dataset index is a combination of event_date and seat location (section, row, begseat, endseat) which is unique for each ticket. This helped spot duplicates and mistakes in the raw data.

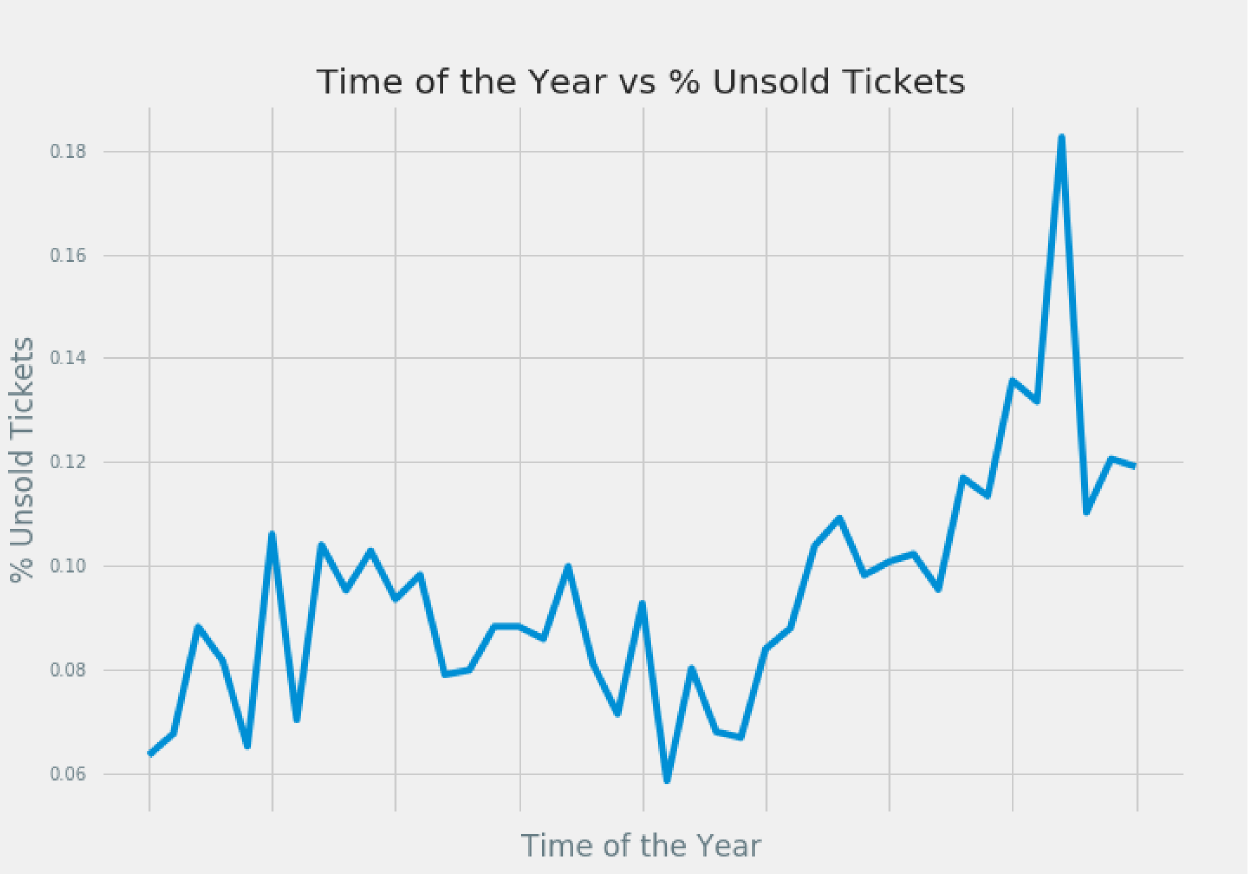
Data Understanding + Data Preparation: Visualizations

Notebook: *Data Understanding (Visualizations).ipynb*

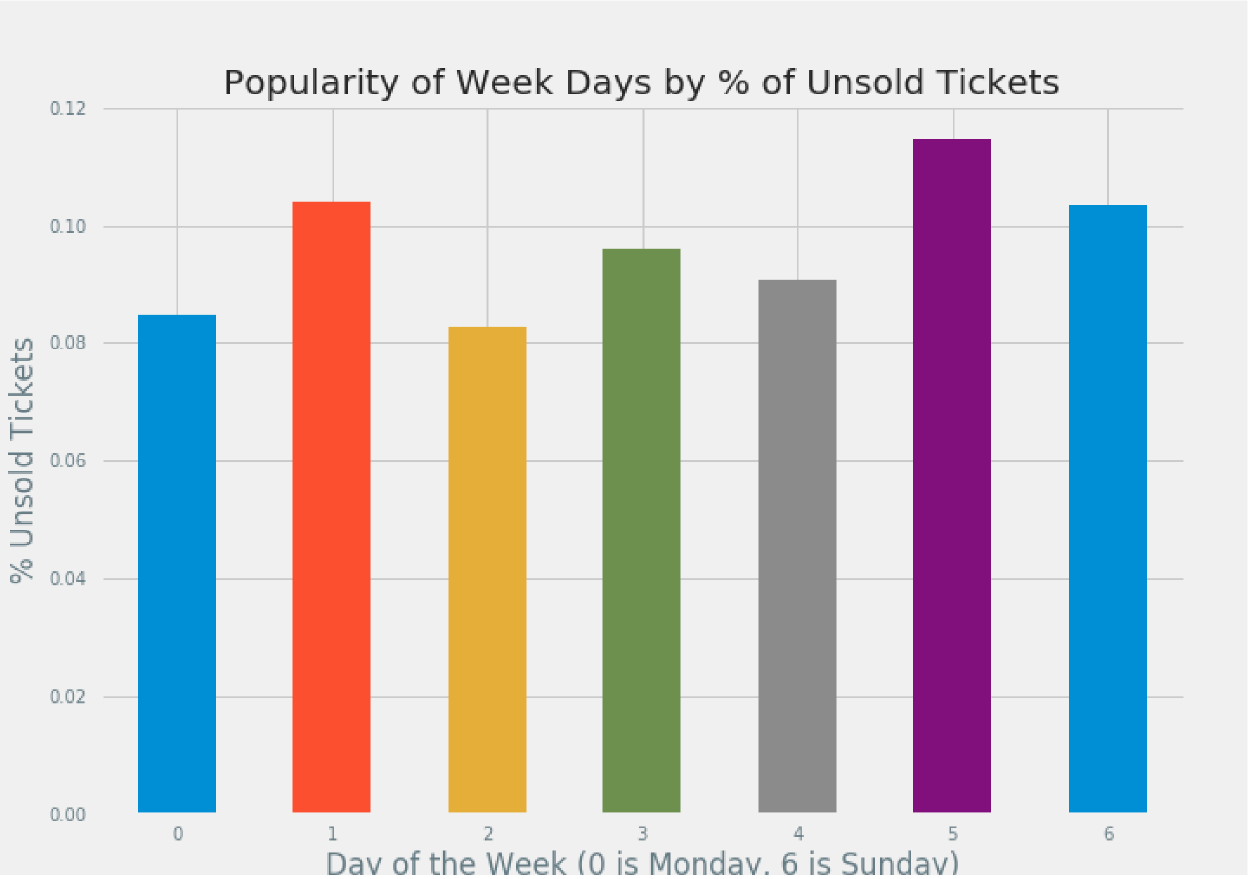
After selecting our features, we wanted to visualize the dataset and examine some of the possible relationships.



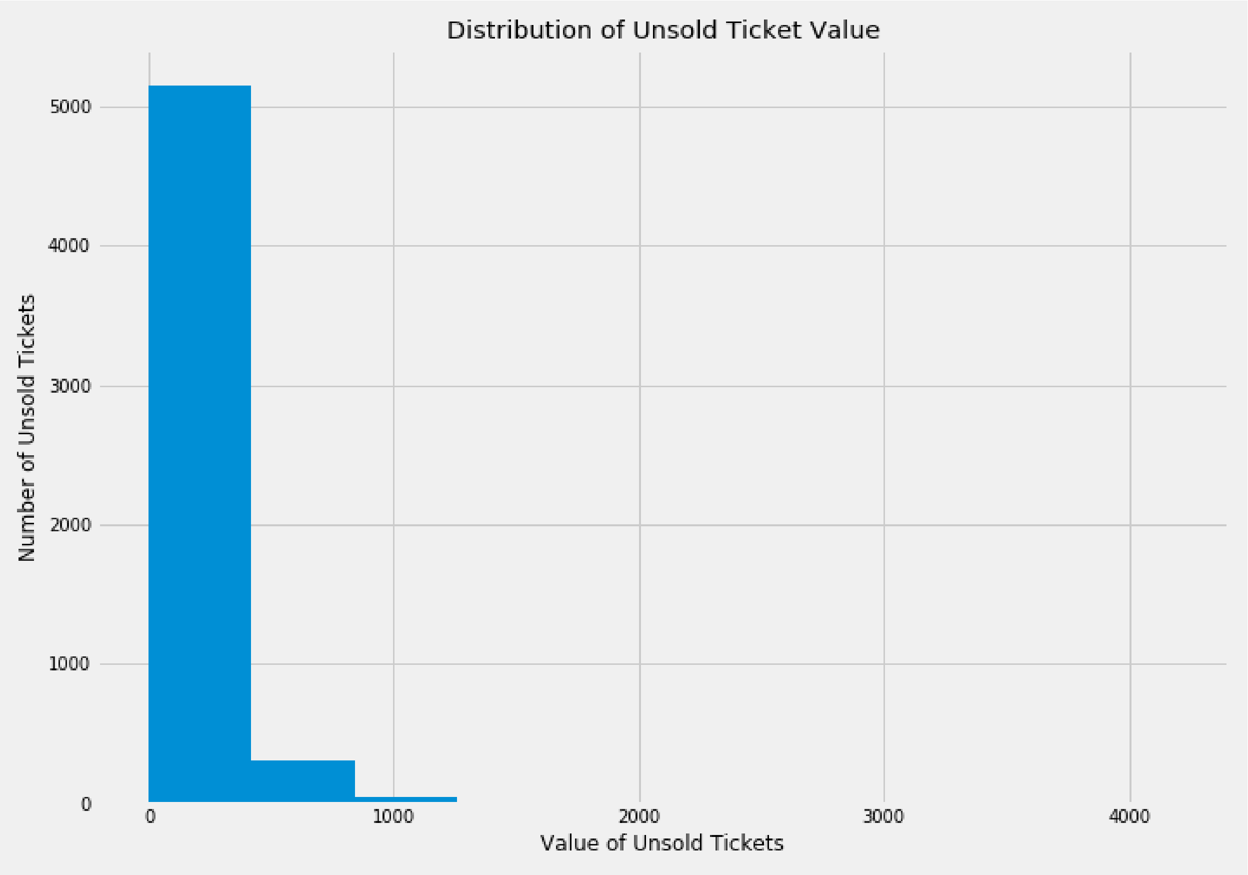
As expected, the % of unsold tickets did vary with each team. The order seems quite reasonable, rebuilding teams such as the 76ers (in 2016) and the Suns had the highest unsold tickets since their matches tend to be less competitive. Popular high performing teams like Warriors and Cavaliers had much lower % of unsold tickets. However, there are some unusual positions like the Wolves, which was an average performing West Coast team with a surprisingly low % of unsold tickets.



The spike in unsold tickets during the end of the season is not surprising at all. The Nets have been struggling to make the playoffs, and around this time of the year, fans will likely either tune out from the Nets or switch their focus to a playoff team. The drop in unsold tickets towards the very end is usually due to the celebratory atmosphere that comes with the last few games of the season, and fans will likely come regardless of the quality of the match.



We were very surprised by the average % of unsold tickets across the days of the week. We expected higher number of unsold tickets during the weekday and lower number of unsold tickets during weekends, but the reverse seems to be happening. We investigated further and found that Nets have the least number of games played on a Saturday and that their “better” matches are usually scheduled for Sunday. Therefore, the % unsold tickets here is not reflective of popularity of Nets games in the weekends. However, we are still surprised by how uniform the bar chart is.



Lastly, the ticket value (total bundle price) is very skewed to the right. This is not surprising as most tickets/bundles are in the $1,000-0 range. We found the distribution to be very much the same for sold tickets and total tickets, which indicates this is not unique to unsold tickets, but rather, the pricing distribution of Nets games in general.

Before we begin to explore the models, we decided to reduce the feature size. In practice, our dataset was 58,418 x 320 due to the dummy variables. From a combination of trial and error and domain knowledge, we were confident that many of the dummies were not informative (many of them were role players that had little influence on attracting fans). We used SelectFromModel to reduce the features to ~100. In addition, we applied a MinMaxScaler to scale down the features.

Modeling

Notebook: *Model and Evaluation.ipynb*

Since the prediction of whether a ticket will be unsold is a classification data mining task, we have tested four different types of models: *Decision Tree Classifier, Logistic Regression, Naive Bayes* and *Random Forest* to approach the problem.

Decision Tree (or classification tree) is a popular model for supervised segmentation by recursively selecting the most informative attributes to split the population to subsets (nodes), and finally reach the leaf nodes that give a probability estimation. We used information gain from minimizing entropy of the data set to rank the ticket features, and started the tree with the attribute test that best split the group. The tree model is easy to interpret and has good overall performance, but it also has a tendency to overfit data easily.

Logistic Regression is another model widely used for estimation of class probability. It is a regression model used in prediction of categorical target variable (usually binary) based on a weighted combination of all attributes. The logistic regression, however, produces a smooth linear decision boundary which may not perform as well on a large dataset with many features, like ours, compared to a tree structure.

Naive Bayes is a very simple but effective method for classification by making a strict assumption that each feature of the instance is independent of each other. The assumption makes the classification results more extreme in the correct direction, which is favorable, when certain features are correlated. However, the actual probability estimation from *Naive Bayes* model is not as useful for our costs and benefits evaluation.

Random Forest is an ensemble method that constructs multiple random decision trees. The idea is that a combination of weak classifiers can form a strong stable classifier. *Random Forest* will select and split on the best feature in a random subset of features, which improves the diversity of the model and reduces the problem of overfitting found in simple tree classifiers. The biggest problem with Random Forest is that it is computationally taxing and slow in deployment. However, we do not anticipate our model to be deployed for real-time predictions.

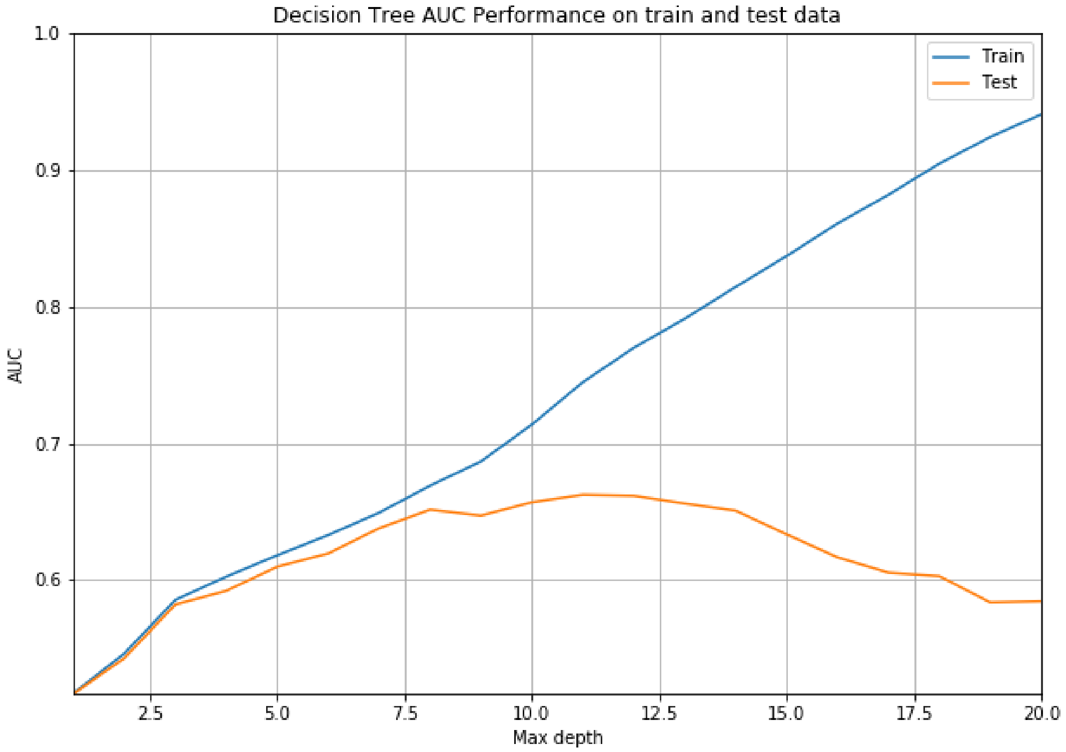
Complexity Control: Fitting Curves

We decided to perform some basic complexity control on the four models before comparing them. This has two benefits— it gives us more generalizable models when comparing performance, and it also simplifies the models and reduces computation time for the rest of the data mining process.

We looked at the *fitting graph* of each model to choose the optimal complexity to prevent overfitting. The graph shows the model performance on the training and testing data as a function of complexity (e.g. number of nodes in tree). Since the models tend to overfit the training data when complexity increases, we selected complexity parameters that gave the best performance before the training and test performance diverged. We used AUC (Area Under ROC Curve) as the performance measure rather than accuracy because we care about the false positive rate, which can impact our expected profit. We do not want a model that gives a high accuracy of predicting unsold tickets at the expense of large false positives (predicting the majority of tickets as unsold, which leads Barclays Center incur unnecessary costs from offering discounts or benefits to tickets that would have sold).

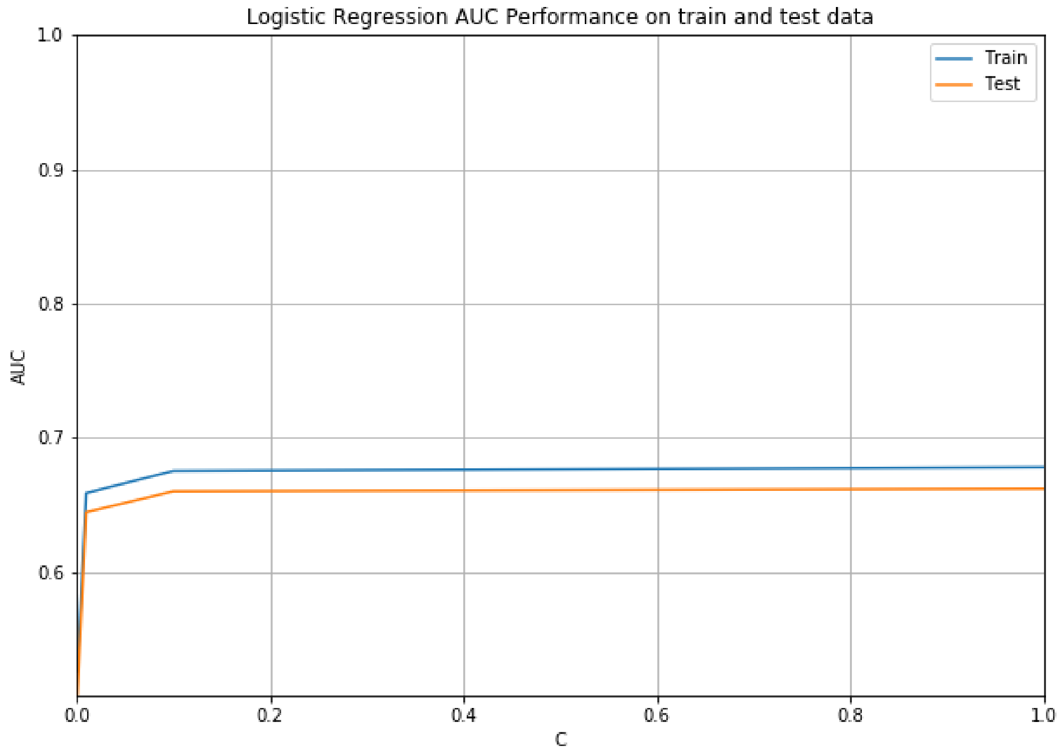
Naive Bayes model doesn't have a measurable complexity parameter. The fitting graphs of the other three models are as follows:

Decision Tree Classifier



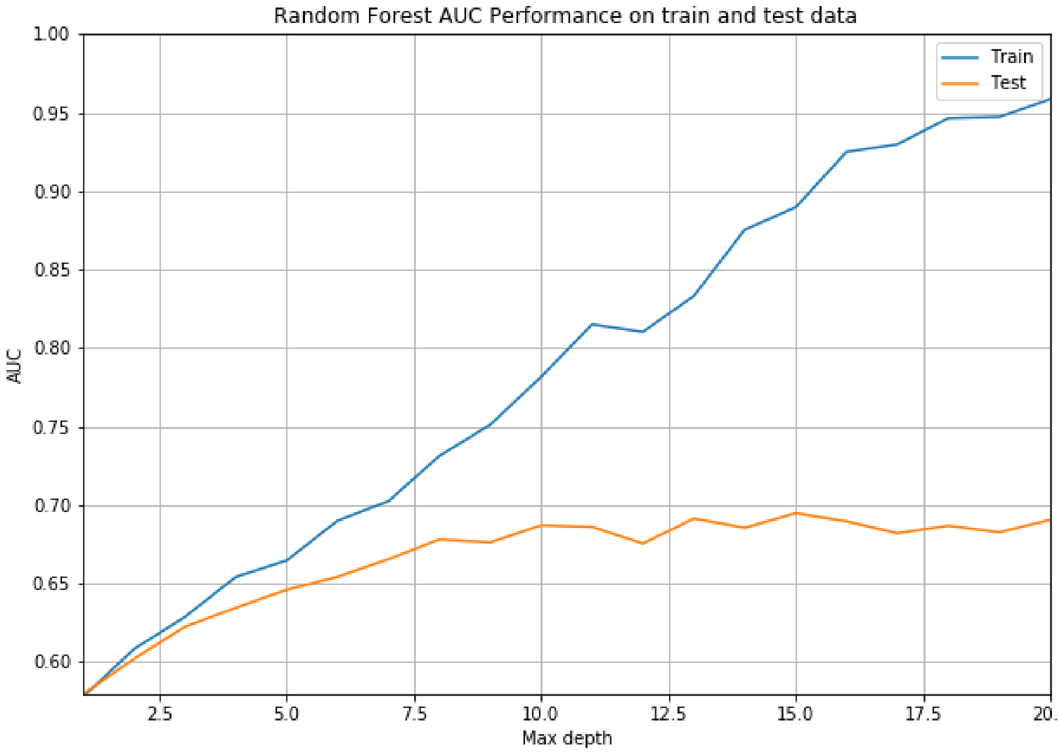
Decision Tree usually has a high tendency to overfit by actually memorizing the features and labels of training data, given increasing branches and nodes and less instances at each leaf. The tree model has great performance on training data, but due to its complexity has a poor generalization performance on holdout data. Therefore, based on the fitting graph, we chose a maximum of 12 tree nodes as the optimal complexity (max depth represents the number of tree nodes).

Logistic Regression



For the logistic regression, there is very little change in AUC from adjusting complexity parameter C. Tuning C performs regularization and would normally affect AUC, but AUC is stable here because we have already scaled the features down with MinMaxScaler during preprocessing.

Random Forest



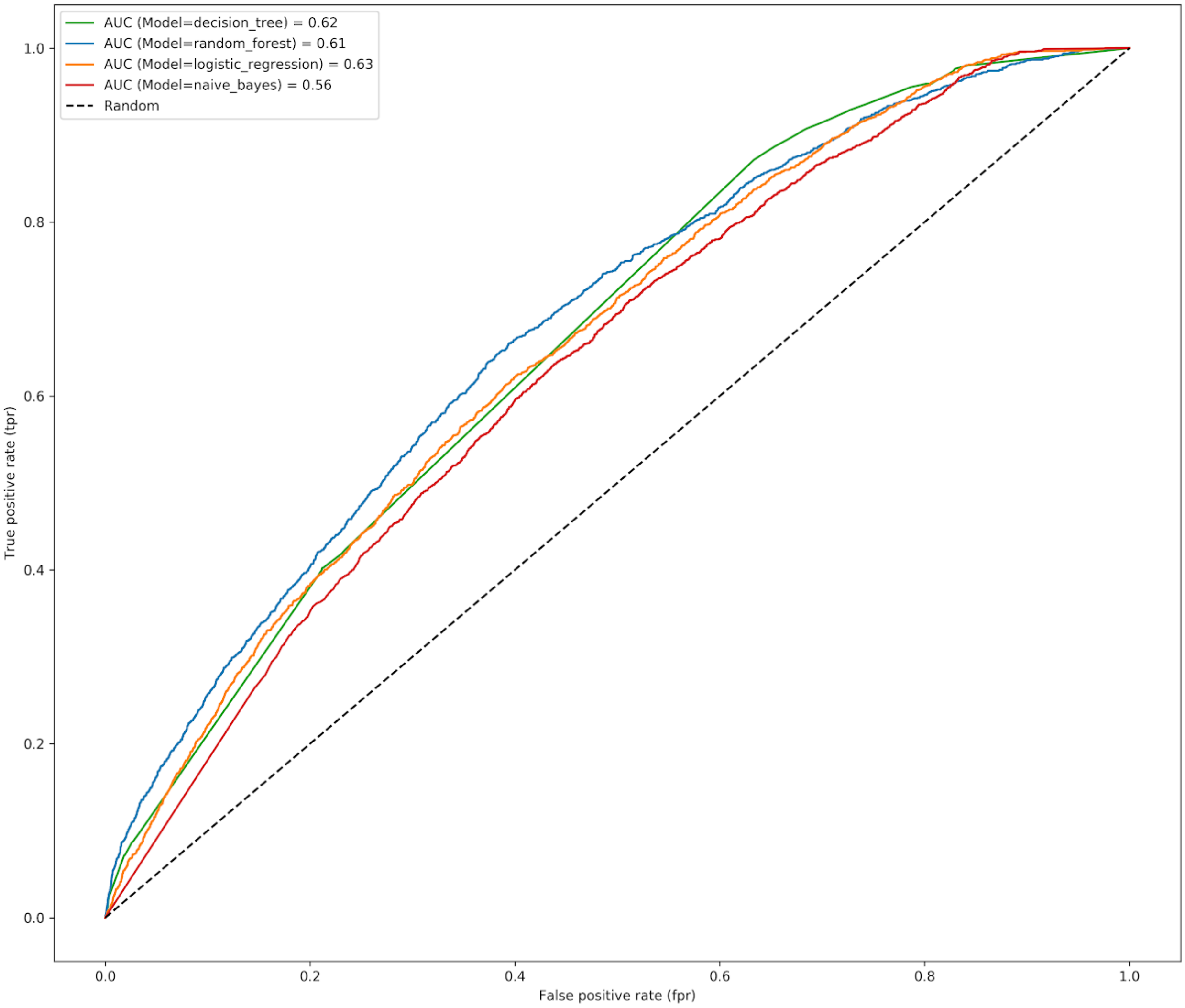
Random Forest is less likely to overfit compared to a simple *Decision Tree* classifier because of its characteristic of randomly choosing the combination of features to split the set rather than strictly following an order. The complexity control is similar, we restrict the number of tree nodes by specifying max depth. The fitting graph of *Random Forest* does not seem to have a distinct "sweet spot" like simple tree model where performance on testing data reach the best performance, but fluctuates and declines when the model gets more complex. We chose the complexity at around 15-20 max depth for this model.

Comparison: ROC and Lift Curves

In order to better compare the four models and understand which model suits our business goal of identifying and reducing the number of unsold tickets, we will examine the *ROC Curve* and *Lift Curve* to visualize the performance of our models.

ROC Curves

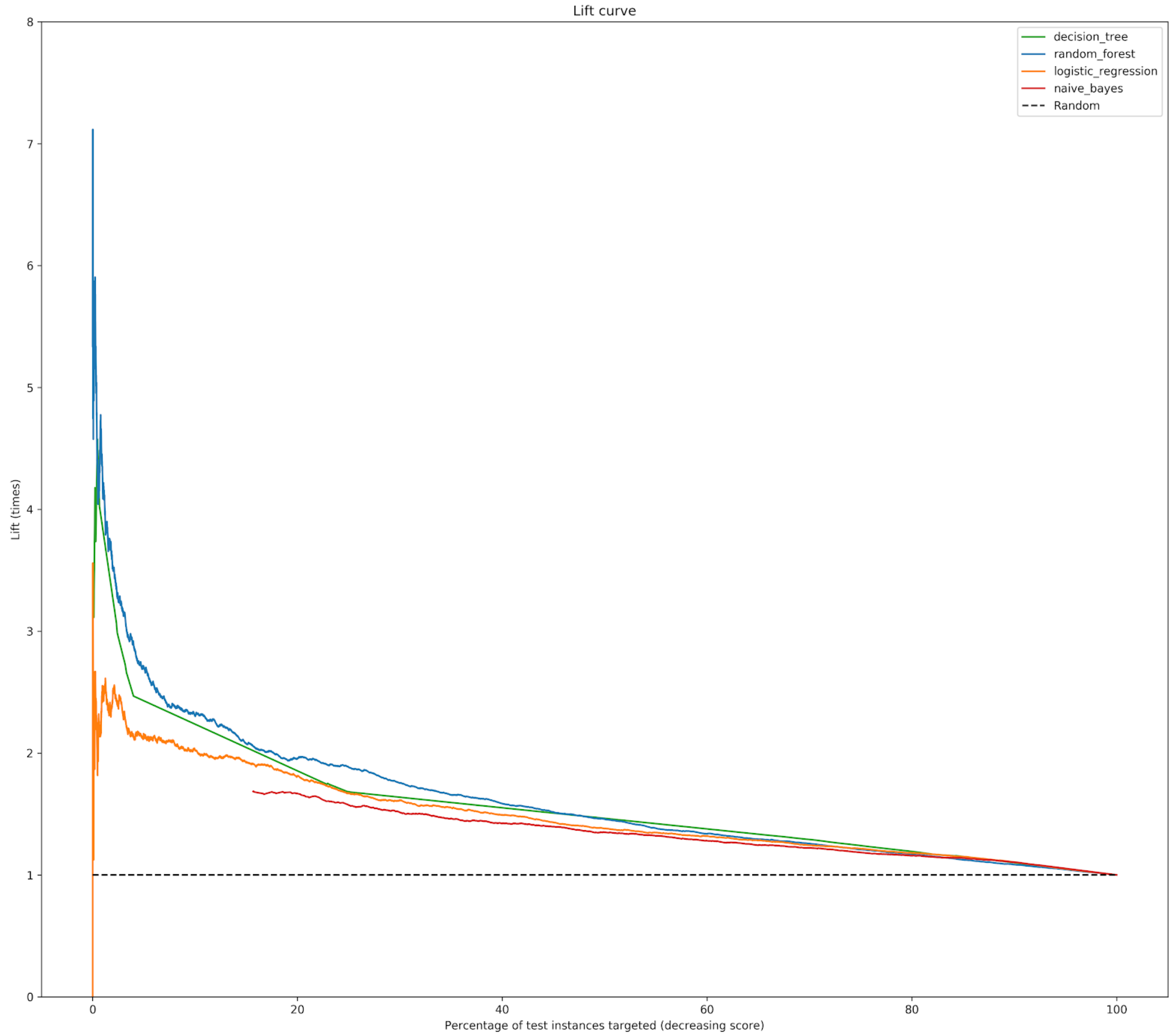
ROC (Receiver Operating Characteristics) Curves are a commonly used to visualize model performance for classification and class probability estimation. It is a two-dimensional graph that shows the tradeoff between a model's false positive rate and true positive rate. We would prefer a model that gives higher true positive rate or makes fewer false positive errors, which lies on the northwest of the coordinate space. We can also easily compare the model's performance with a baseline model, which randomly predicts a ticket to be unsold and is represented by the dotted y=x line.



In our business scenario, a true positive condition is when the model predicts a ticket to be unsold and it is actually unsold, while a false positive condition is when the ticket is predicted to be unsold but is actually sold. As is shown in the ROC graph above, all four models we have built perform better than randomly guessing the positive classes, which is similar to broad marketing campaigns that Barclays Center may undertake to offload unsold tickets. The area under the curve (AUC) for all of the models is greater than 0.5. Among the four models, *Random Forest* has the best performance in true positive prediction when false positive rate is less than 0.6, while *Decision Tree* predicts higher true positive rates when false positive rate is greater than 0.6. As is mentioned before, we do not want a high false positive rate because falsely predicting unsold tickets will cost us high expenditures in unnecessary marketing cost. Therefore, *Random Forest* seems the best model with lower false positive rate and higher true positive rate.

Lift Curves

Lift can be calculated as the percentage of true positives over the base rate of positives, in other words, an estimate of how much better our models perform compared to randomly selecting instances to find positives.



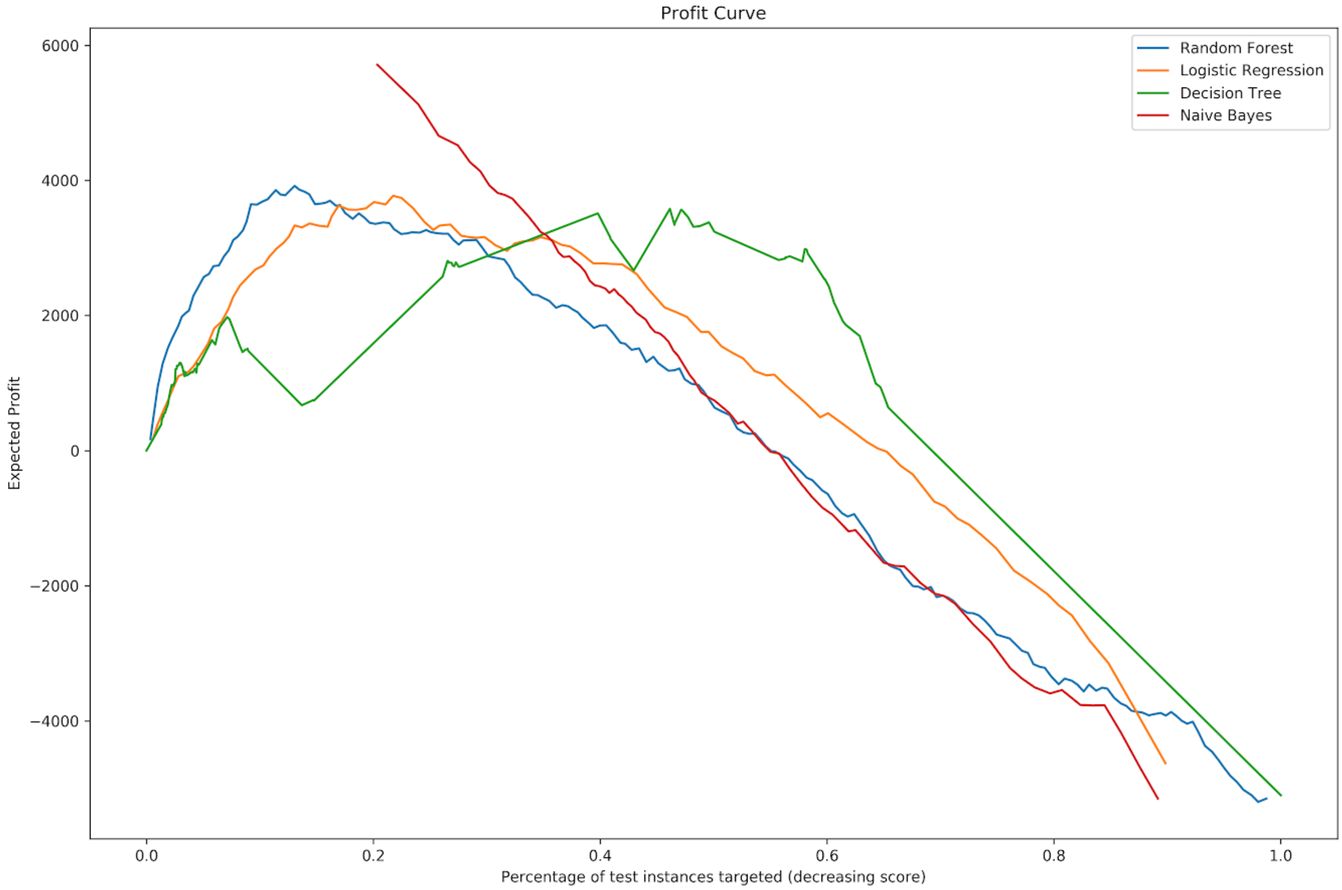
Our models clearly performs better than selecting randomly. *Random Forest* had the highest lift when targeting a low % of instances, above 7x lift. The *Decision Tree Classifier* also performs quite well and maintains a lift slightly below Random Forest for low to medium % of instances targeted. Both *Logistic Regression* and *Naive Bayes* have positive lift but lower than the other two models.

Evaluation

While the ROC and Lift Curves provide us a good understanding of how the models compare to each other, we decided to select and evaluate our model through the expected value framework as it best reflects the full cost benefit of our proposal. As mentioned at the start, we expect Barclays Center to act on tickets/bundles that are predicted to be unsold. To apply the expected value framework, we assumed Barclays Center will incur additional marketing cost to convert unsold tickets into sales, at $2.00 per ticket, and that the marketing campaign will always be successful. Hence, our $b(Y|p)$ = Revenue of the Ticket - $2.00 for each correctly predicted unsold ticket. For false positive predictions, $b(Y|n) = -\$2.00$ for each ticket. Therefore, our expected profit for each ticket can be expressed as:

$$\text{Expected Profit} = p(p) * p(Y|p) * (\text{Revenue of the Ticket} -2.00) + p(n) * p(Y|n) * (-2.00)$$

This cost/benefit matrix is interesting as the revenue of each ticket can be different, hence, we had to compute the expected profit per ticket and sum to the total expected profit. Below is the code we used to perform this calculation and generate a profit curve.



The profit curve shows *Naive Bayes* has the highest possible expected profit near $6,000. This is likely due to the fact that *Naive Bayes* tend to over or underestimate probabilities, improving its true positive rate and expected value. However, we are concerned that the *Naive Bayes* model predicted 0 unsold instances between approximately 0-20%. This could be a problem in deployment where the percent targeted is near a threshold in which the *Naive Bayes* predicts every ticket as sold, and fail to address our business problem. The *Decision Tree Classifier* performed poorly when we targeted a low percentage of test instances, but outperforms other models at a higher percentage of test instances. However, we expect to limit our targeting to a low % of instances as this proposal is intended to be a low budget solution, and executing a marketing campaign for 50-60% of all tickets could require a larger starting investment. The Logistic Regression and Random Forest Classifier performed quite similarly, with Random Forest being the best model to target a low % of instances, while Logistic Regression performing fairly average across all % of targeting.

We decided that the Random Forest model is most suitable to reach our business goals. As mentioned, our solution is intended to generate profits over a low budget. The Random Forest performed the best with a small % of instance targeted at around 10-20%, where it achieved the highest lift at about 7.3. The maximum expected profit from this model is $4,000. The success of this project can be measured as the actual profit generated divided by the investment required to deploy and maintain the model.

Deployment

The model is designed to be deployed in the 1 week period before the actual game day. We do not expect this model to be used in real-time to monitor opportunities to offload unsold tickets, but the features required should be readily available and easy to access. The model can be integrated with internal systems (such as the company database) to predict unsold tickets, perhaps once a week. The output should be the graphs we have discussed throughout this report— ROC, Lift and particularly Profit Curve, as well as actual numbers for expected profit. The profit curve should work well because the priors will likely remain the same. The Nets, despite poor performance, is still a very popular due to its location in Brooklyn and near New York City, hence the low base rate for unsold tickets. We do not expect the unsold tickets to increase. Likewise, if Nets become much more popular, there is limited room for increase in tickets sold. Hence, the priors should be stable.

Some adjustments may need to be made over time as the model is deployed. One problem we encountered is the impact of new players. The model looks at recent historical data, and that tends to adjust well for typical players that join the league and improve over time, increasing their influence on ticket sale. However, it is not uncommon for rookies to have exceptional first-year performance and rapid growth in popularity (Donovan Mitchell in 2018-17 season). Therefore, we recommend Barclays Center to look into incorporating social media features, or even pre-draft statistics to account for these players. It even be necessary to adjust the data, such as expanding the list of All Stars to unofficial All Star caliber players. That can allow for injection of domain knowledge if Barclays Center believes a certain player can attract significant ticket sales (e.g. Jeremy Lin, he has a large fanbase but his stat sheet may lower his influence).

Lastly, the current proposal and expected profit calculations assumes a fixed marketing cost to convert unsold tickets to sold tickets. The first risk is that the marketing is significantly less effective than we assumed, in which the expected profit will lower. The second complication is that Barclays Center will likely be more price competitive if they applied a discount model instead, where they slash prices on predicted unsold tickets. However, the risk here is that the false positive cost becomes much larger, since it is now a discount offered to a customer whom would have purchased the ticket anyways. This is particularly risky if the model identifies an expensive ticket as unsold, and the discount is large in dollar terms. Barclays Center may need to improve the model significantly in order to benefit from a shift to a discount model, but that is difficult given the base rate.

Member Contributions

Horace Fung is the seed of our group so he worked very hands-on with the data itself. His technical knowledge was crucial in preparing the data, running the models and evaluating the results. Amber Wang worked to read and understand the models, comparing them all and using classroom knowledge to see the advantages and disadvantages of each and interpret their results. Nicole Rodriguez looked at the models with a more business-minded perspective, evaluating the models more generally and seeing how their results tied back to our original issue and affect the business model and the company's profit/loss. Dylan Prada currently works for the Brooklyn Nets and provided us with the data and context we needed to make the models. His industry knowledge gave us a more in-depth and contextual perspective on the data and results, allowing us to better narrow down and clean the data, predict informative features and make conclusions about the results. Overall, from the initial proposal to the final report, all members contributed fairly and collaborated to present to you our final project.