Order Allocation System Change Evaluation Report

Objectives

- **Change Objective:**
  To improve order match time; that is, the duration since the time an order was created to the time the order was accepted by a driver
- **Evaluation Objective:**
  To see if the impact of the change introduced on ‘2017-03-30 12:00:00 AM’ is statistically significant based on the test results; so we get to decide if we should have a full rollout of the new system

Report Rundown

The whole evaluation report consists of three sections, which cover three aspects regarding the data set and test results.

1. **Resulting Impact:**
   This section aims to answer the core question - whether the change drove a significant result
2. **Insights Discovered:**
   This section reports additional insights discovered based on the gathered data
3. **Suggestions and Improvement:**
   This section includes ideas/ thoughts for system and experiment design in the future

Dataset

The original data set consists of 5000 observations of orders created from 2017-03-24 to 2017-04-03. Each observation has the following information (Variables):

- **Driver Response Timestamp**: This is the time when the driver response to an order
- **Order Create Timestamp**: This is the time when an order is created by a customer
- **Order ID**: The identification number of an order

In order to conduct a thorough analysis based on these, the data set was manipulated and preprocessed before any sorts of analyses. We further extracted information based on the above features, and created additional variables for each observation:

- **match_time**: This is the order match time in xx:xx:xx format. That is, the time difference between Order Create Timestamp and Driver Response Timestamp
- **total_seconds**: This is the order match time formatted in seconds
- **order_time_lag**: This is the time difference between the current order (t) and the previous order (t-1)
- **day_of_week**: This is the day when the order was created (eg. Monday, Sunday)
- **session**: This is the session of a day. A day is divided into four sessions based on time intervals
  - Night - Between 00:00 and 06:00
  - Morning - Between 06:00 and 12:00
  - Afternoon - Between 12:00 and 18:00

- Evening - Between 18:00 and 00:00

Technologies

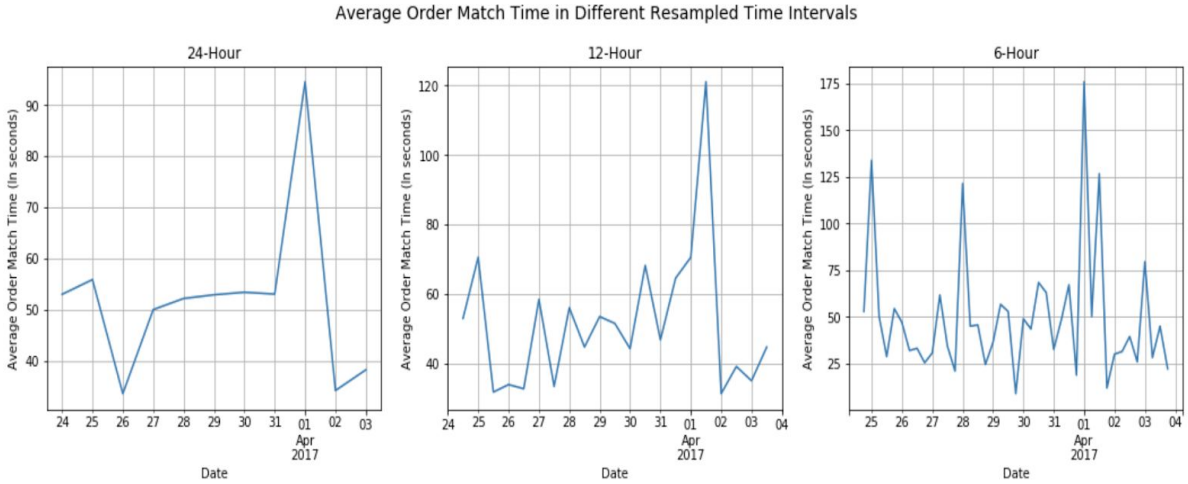
All analyses were done within a Python 3 environment. Additional scientific computing, statistical, and machine learning packages are required to reproduce the results. Please refer to the supplementary iPython notebook for details.

****

1. Resulting Impact

1.1. Rationale & Background

The approach we took to evaluate the resulting impact was to conduct A/B testing. Essentially, we conducted a hypothesis testing on a test statistic calculated from the data set to see if there was a statistically significant result, namely, the new system improved order match time. High level exploratory data analysis was conducted beforehand, and here is a visualization on average order match time through the entire test period:



Observations were resampled and aggregated to smoothen out noise. As shown in the visualization, there seemed to be certain spikes after 2017-03-30. Nonetheless, this is not meant to be probabilistic; we needed further investigation to see if the new system was useful or not.

1.2. Highlighted Techniques

- Hypothesis Testing
- Bootstrapping
- Permutation

1.3. Implementation

In order to perform hypothesis testing, we split the data set into two using the date we introduced the system as a threshold. Basically, any orders created before 2017-03-30 were old-system orders (Control group); whereas others were new-system orders (Treatment group). The test statistic we used was the difference of average order match

time (In seconds). The significance level was 0.05, and the corresponding confidence level was 95%. Here are the statements of the hypothesis testing:
Null (H0): average match time (new-sys) - average match time (old-sys) >= 0
Alternative (H1): average match time (new-sys) - average match time (old-sys) < 0
This could also be written in:
Null (H0): average match time (new-sys) >= average match time (old-sys)
Alternative (H1): average match time (new-sys) < average match time (old-sys)
For getting a probabilistic and statistical result, we leveraged permutation and ran simulations for 10000 times. Permutation is a technique to perform random reordering of entries in a data set. In our test, observations were therefore getting randomly re-assigned to one of the two groups; hence the average calculated for each group in each simulation would be different. In our evaluation, here are the actual steps that we went through:

1. Concatenated the two groups
2. Performed random reordering on the concatenated data
3. Split the data into two groups
4. Calculated the difference of average order match time
5. Simulated the entire process 10000 times

1.4. Results

This process helped us generate a sampling distribution shown below:

```
Empirical Average Order Match Time Difference = 2.15718489659951
95% confidence interval = [-11.66695783  11.85456628]
p-value = 0.6425
```

The visualization above is good enough to answer the question. There are two points to note:

1. The p-value is no less than the chosen significance (alpha) level
2. The confidence interval contains the null hypothesis value

In academic vernacular, p-value is the probability of obtaining a value of the test statistic that is at least as extreme as what was observed, under the assumption the null hypothesis is true. On the other hand, confidence interval means if we ran simulations over and over again, p% of the observed value would lie within p% confidence interval.
In our case, the null value (0 as there is no difference) lies within the 95% confidence interval. And such a large p-value (0.6425) suggests that there is no statistical significant difference shown that we should reject our null hypothesis; that is, we'd expect to see sample as extreme as our observed value for about 64% of the time assuming the null hypothesis is true. In other words, we do not reject the hypothesis that the new-system orders had an average order match time greater than/ equal to the old-system orders'.

1.5. Extension

We actually went beyond and reframed the problem as a conversion problem. We wanted to create a label for each order to indicate whether the order was a 'fast-grab' order. In order to do so, we needed a benchmark/ threshold to help us decide. We ended up choosing the average order match time based on the entire data set. The benchmark value was chosen probabilistically using bootstrapping. Bootstrapping estimates the variability of the sampling process; so it actually works better than permutation does for estimating confidence intervals, which is essentially a range of values where our benchmark value is likely to lie within. In a nutshell, bootstrapping is a technique used to resample data to perform statistical inference. Here are the steps we went through for getting the benchmark value:

1. Resampled the dataset; that is, sampling from the data set with replacement
2. Calculated the average order match time
3. Simulated the entire process 10000 times

The process helped us generate a sampling distribution shown below:

```
Empirical Average Order Match Time = 48.8912
95% confidence interval = [43.312575 54.95149 ]
```

The visualization shows that the observed value, which is the average order match time of the entire data set, lies within 43.31 and 54.95 seconds; that is, 95% of our observed values would lie within this range if we keep simulating the process. This makes the current observed value a good estimate for our benchmark value. We created an additional binary variable named **'fast_grab'** to indicate whether the order was responded less than 48.8912 seconds.

The test statistic we used was the difference of conversion, which is just the difference of the proportion of 'fast_grab' within each group. The significance level was 0.05, and the corresponding confidence level was 95%. Here are the statements of the hypothesis testing:

Null (H0): conversion (new-sys) - conversion (old-sys) <= 0

Alternative (H1): conversion (new-sys) - conversion (old-sys) > 0

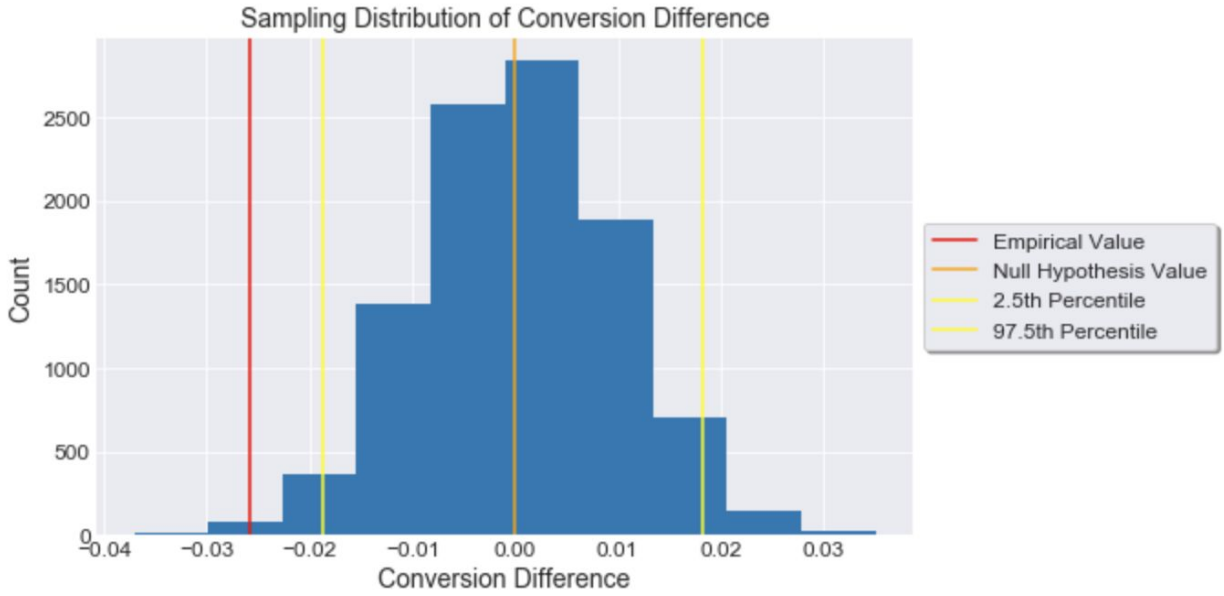
This could also be written in:

Null (H0): conversion (new-sys) <= conversion (old-sys)

Alternative (H1): conversion (new-sys) > conversion (old-sys)

Again, we used permutation to run simulations, and here is a visualization of the results:

```
Empirical Conversion Difference = -0.02584739814588688
95% confidence interval = [-0.01863632  0.01822032]
p-value = 0.9954
```



The p-value is even larger, and the null value again lies within the confidence interval; therefore, the test aligned with the one done on average order match time.

1.6. Conclusion

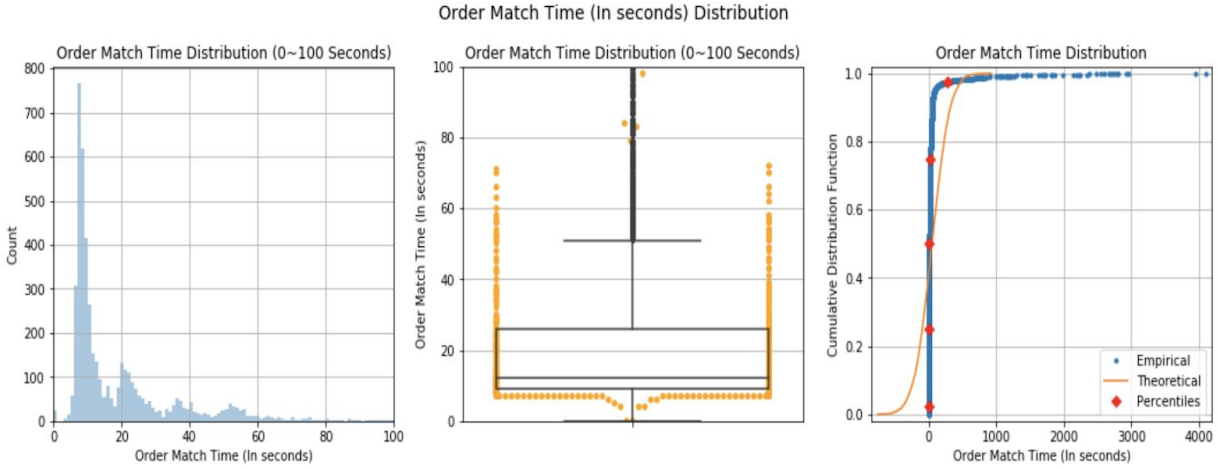
Based on both test results, we concluded that we do not reject the null hypothesis; that is, the new system cannot improve order match time. In other words, there is no statistical significant evidence shows that the new system is proved to be useful in the sense of improving order match time.

2. Insights Discovered

2.1. Rationale & Background

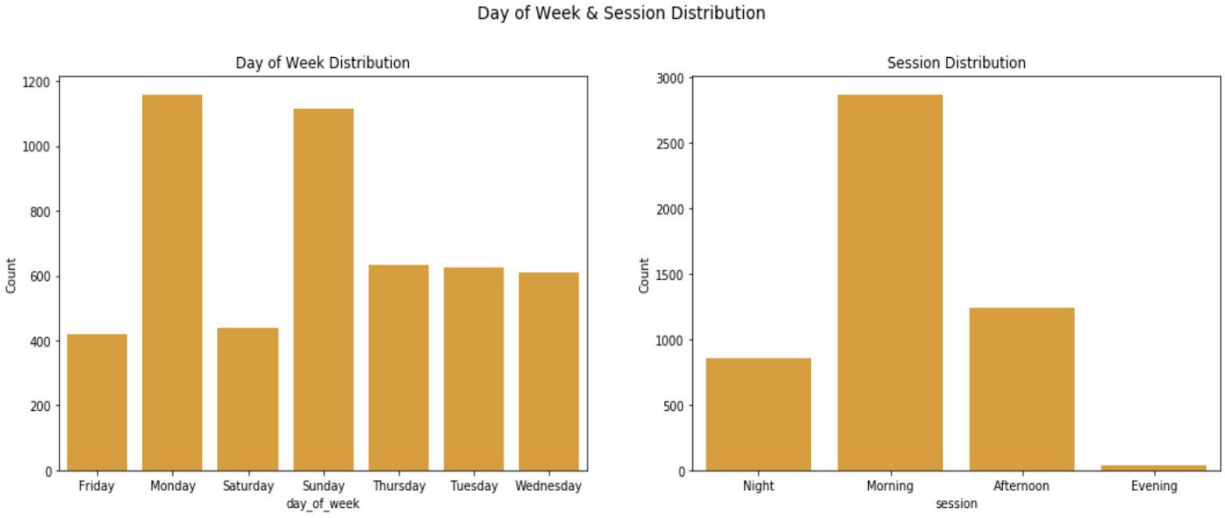
This section includes interesting findings that our team could look into in the future. To caveat, the information (Variables) contained in the data set is relatively limited and small, not to mention the size of the data set itself; so the insights discovered might not necessarily be representative. On top of that, most of the insights mentioned in this section are time-related (Based on the data set context).

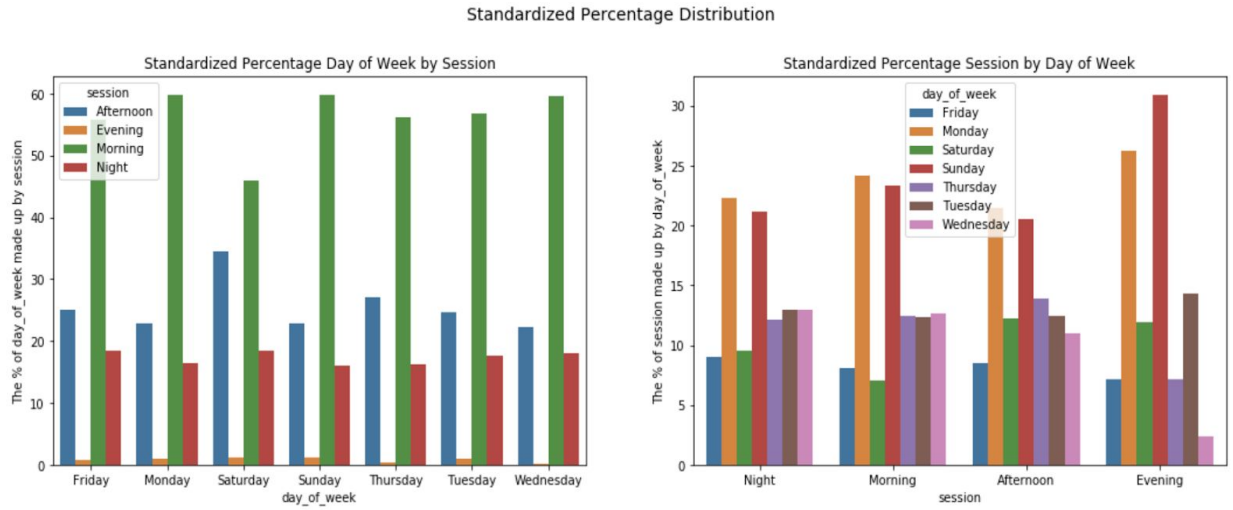
2.2. Distribution of Order Match Time



This visualization shows the distribution of order match time of the entire data set. Note that the histogram on the left and the boxplot in the middle are trimmed (In seconds) in axes for better clarity. We can tell distribution is skewed due to outliers, which are clearly denoted by the cumulative distribution function on the right. About ¾ observations clustered below 30 seconds. It might be interesting to look into the characteristics of these customers as well as the outliers to create better customer segmentation.

2.3. Day-of-Week & Session Distribution

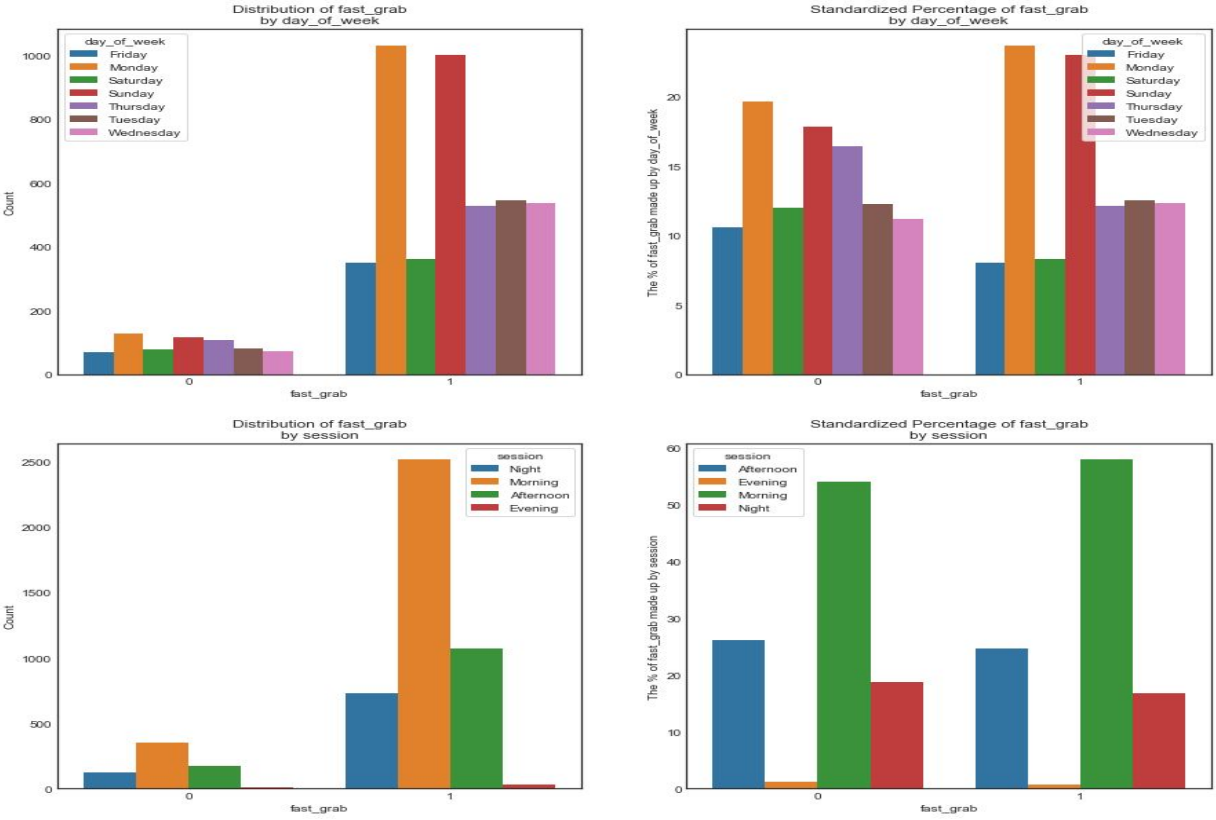




The first visualization shows the absolute value count on orders took place on each day and each session, and the second shows the standardized percentage (Proportion) made up by each other. It seems that Monday, Sunday, and Morning are generally good indicators of when an order takes place; whereas Friday and Evening seem to be bad indicators. This piece of information might be useful from an operations' perspective as team members could allocate resources more effectively for certain periods.

2.4. Relationship Between Fast-Grab, Day-of-Week, and Session

As shown in the previous section, some days and sessions seem to have greater/ fewer number of orders. We wonder if this applies to a narrower scope (certain types of orders):



Monday, Sunday, and Morning seem to be have higher association with 'fast_grab'. We leveraged Logistic Regression to model the data to see if certain days or sessions have an higher/ lower association with 'fast_grab' over others. Here is a summary of the model:

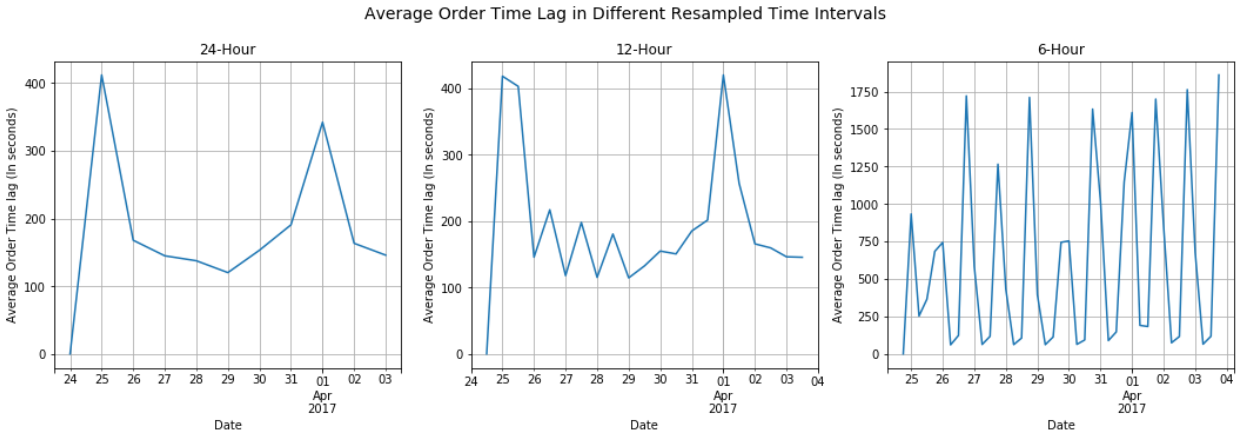
```
                           Logit Regression Results
==============================================================================
Dep. Variable:              fast_grab   No. Observations:                 5000
Model:                          Logit   Df Residuals:                     4990
Method:                           MLE   Df Model:                            9
Date:                Mon, 02 Jul 2018   Pseudo R-squ.:                0.009249
Time:                        12:59:22   Log-Likelihood:                -1915.9
converged:                      True   LL-Null:                       -1933.8
                                       LLR p-value:                 4.350e-05
============================================================================================
                               coef    std err          z      P>|z|      [0.025      0.975]
--------------------------------------------------------------------------------------------
day_of_week_Monday           0.4627      0.162      2.860      0.004       0.146       0.780
day_of_week_Saturday        -0.0804      0.182     -0.442      0.659      -0.437       0.276
day_of_week_Sunday           0.5306      0.164      3.228      0.001       0.208       0.853
day_of_week_Thursday        -0.0299      0.169     -0.177      0.860      -0.362       0.302
day_of_week_Tuesday          0.2960      0.178      1.662      0.097      -0.053       0.645
day_of_week_Wednesday        0.3681      0.182      2.028      0.043       0.012       0.724
session_Night                0.3776      0.407      0.927      0.354      -0.421       1.176
session_Morning              0.5412      0.399      1.355      0.175      -0.241       1.324
session_Afternoon            0.4486      0.404      1.111      0.267      -0.343       1.240
const                        1.1390      0.415      2.744      0.006       0.325       1.953
============================================================================================


The Odd Ratio and Corresponding Confidence Interval (95%)
                             2.5%      97.5%  odd_ratios
day_of_week_Monday       1.156711  2.180981    1.588321
day_of_week_Saturday     0.646058  1.318027    0.922779
day_of_week_Sunday       1.231676  2.346088    1.699888
day_of_week_Thursday     0.696579  1.352240    0.970537
day_of_week_Tuesday      0.948293  1.906233    1.344495
day_of_week_Wednesday    1.012351  2.062539    1.444996
session_Night            0.656702  3.240227    1.458719
session_Morning          0.785526  3.757732    1.718079
session_Afternoon        0.709828  3.455614    1.566171
const                    1.384629  7.047312    3.123766
```

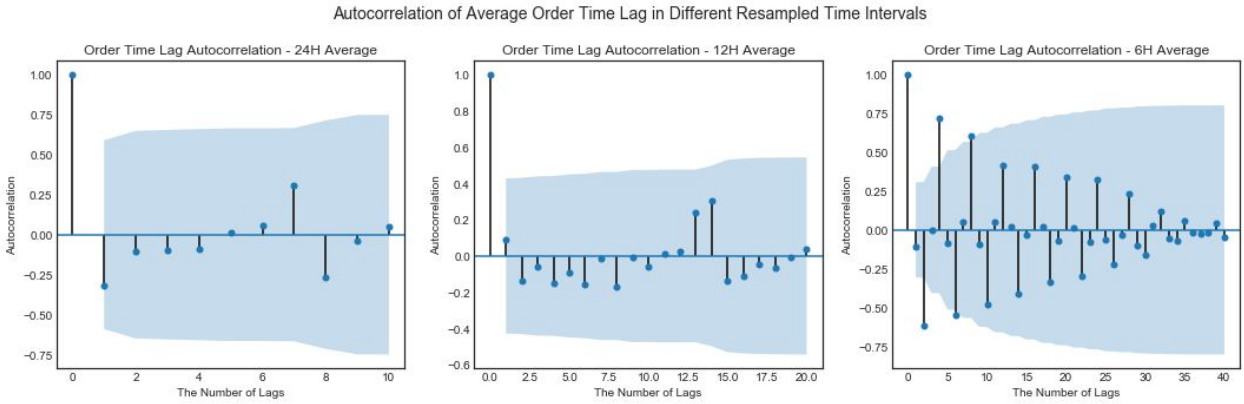
Assuming alpha = 0.05, the coefficients of Monday and Sunday seem to be statistically significant. With Friday being the baseline, Monday, Wednesday, and Sunday seem to be days more associated with 'fast_grab' orders, namely, orders that get responded within benchmark value (Around 48.89 seconds). Looking at the odd ratios, if an order took place on Monday, Wednesday, or Sunday, it'd be about 1.59, 1.44, and 1.70 times respectively more likely to be a 'fast_grab' order than if it took place on Friday, holding all other variables constant.

2.5. Order Time Lag & Autocorrelation

One of the things that caught our interest upon receiving the data set was the time lag between each order. We wonder if it would be possible to leverage certain statistical models to make forecast on when the next order might take place. We treated the data set as a time series data, and here is a visualization of average order time lag, resampled in 24-Hour, 12-Hour, and 6-Hour respectively:



The trend got progressively nosier as we up-sampled the data set. We went ahead and plotted the average order time lag's autocorrelation function, which is basically to see if it is correlated with any of the lagged copies of itself (The same time interval). Here is a visualization:



At first glance, it seems there are certain significant autocorrelations when the data is resampled in 6-Hour. Nonetheless, when we ran a regression test to see if the average order time lag is a random walk, we got a p-value of 0.446. We failed to reject the null hypothesis that the average order time lag is a random walk, which basically means there is no concrete way to forecast/ model the next order.

Having said that, it'd be worth trying again when we get to possess a larger amount of data. Being able to predict/ forecast the next order is useful from both customers' and businesses' perspective.

2.6. Supervised Learning Classifier

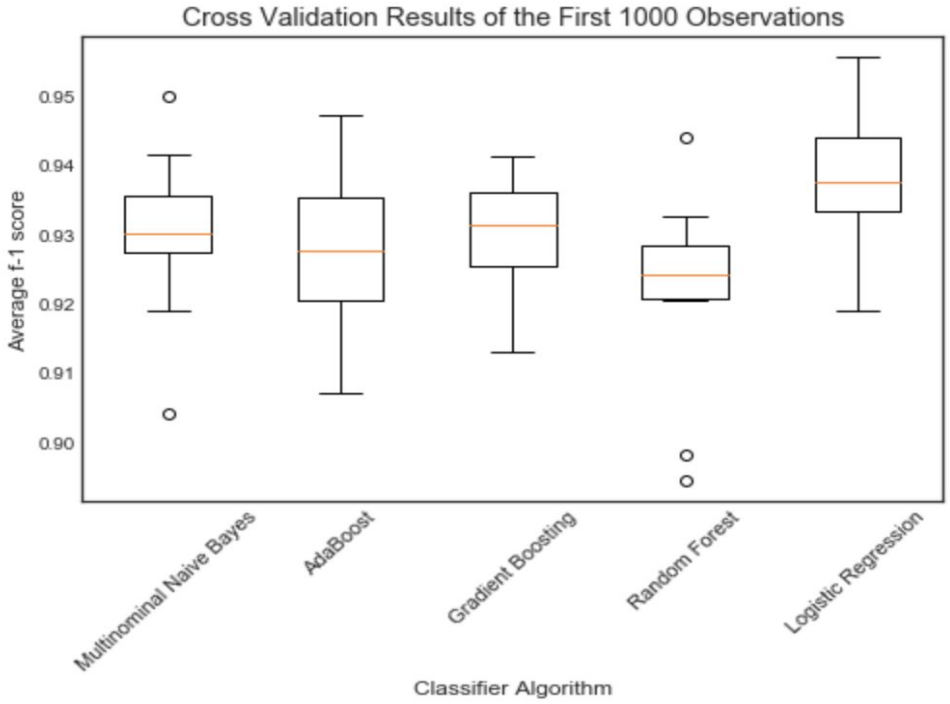
As we found certain variables like 'day_of_week' and 'session' in the data set to be somehow indicative on the occurence of certain orders, we started to ponder if we could leverage machine learning techniques to make predictions. Although we couldn't make forecast from a time series perspective, we could still make use of these variables to frame a supervised learning problem, as we had already labeled our data ('fast_grab') based on a benchmark value. Regarding the features needed to be fed into the model, we basically did the same thing like we did when preprocessing the data, but went further.

We not only treated ‘day_of_week’ and ‘session’ as category variables, but also the ones below:

- **Month**: 1 ~ 12
- **Day** : 1 ~ 31
- **Hour**: 00 ~ 23
- **Minute**: 00 ~ 59

Together with ‘day_of_week’ and ‘session’, we one-hot-encoded all these categorical variables so they can be fed into the algorithms chosen. Here is a visualization of the cross validation results on 1000 observations of the train set:

```
The average f-1 score and standard deviation of each model is:
Multinominal Naive Bayes: 0.9299 (0.0118)
AdaBoost: 0.9285 (0.0115)
Gradient Boosting: 0.9298 (0.0086)
Random Forest: 0.9213 (0.0142)
Logistic Regression: 0.9378 (0.0099)
```



Although this is a high level evaluation, it still surprised us. Without many other features regarding orders, all these model were still able to tell whether orders were ‘fast_grab’. Timing definitely plays a significant role in getting orders matched; so we went ahead and tested these models on a test set. Here are the results of the Logistic Regression model, which was the best performer based on the cross validation results:

```
The Logistic Regression model performance:
                precision    recall  f1-score   support

Not Fast-Grab       0.50      0.02      0.04       159
    Fast-Grab       0.87      1.00      0.93      1091

  avg / total       0.83      0.87      0.82      1250
```

To caveat, this model was trained on 3750 observations, and tested on 1250 observations. This is a small data set. In order to justify the performance, we may need more data as well as considering feeding more relevant features into the model. But the results above suggest the importance of timing, and should serve as a good starter idea which is worth being further developed. Being able to tell whether an order is 'fast_grab' could potentially help us enhance operations efficiency, as we could allocate resources in a more appropriate manner.

To add, we also tried to frame a regression problem, in which we aimed to predict the 'total_seconds' value (Order match time) based on the same set of features with and without 'order_time_lag'. The results, however, was poor as expected. More features or other techniques may be needed for modelling the current setting.

3. Suggestions and Improvements

3.1. Rationale & Background

This section consists of ideas/ suggestions on what could be done to enhance the quality of the experiment. That is, what we could possibly do to improve the testing if we were to do it again in the future. To be specific, although we said that there was no statistical significant evidence showed that the new system was proved to be useful in the sense of improving order match time, we did not mean we should chuck the new system or the entire idea. We might have missed important factors during the process, or even before we started it. In a nutshell, A/B testing is not almighty and perfect, and we should always take other factors into consideration.

3.2. Design (More controllable)

The design of experiment varies with situations. Regarding the current experiment, points that we should ponder include:

- **Customer Segmentation & Multi-variation:** This is basically a question of who the change is for. As shown in previous sections, there are certain patterns among orders. This could also mean certain customers/ operators tend to place/ take orders during certain period of time. Being able to tell how users are characterized helps us hypothesize appropriately, tailor different tests, and perform multivariate analysis/ testing, which could potentially yield more meaningful results.
- **Change Magnitude:** If the change is supposed to be universal, we'd need to make sure subjects are representative enough. For instance, if our new system is meant to take care of all sorts of customers, a random, representative, and large

enough sample size is needed. Currently, we have neither the information of how customers/ operators got picked nor customers/ operators user details. To a certain extent, we were shooting blind.

- **Test Metrics:** There are usually more than one metrics to measure the performance of a change. For instance, although the current objective is to improve order match time, customer satisfaction/ rating could potentially be another metric to measure the success of a change. The point here is we could think of more metrics to capture before starting the process. This could help enhance objectivity.
- **Testing Time/ Period:** This is as much a science as an art. This is one of the most common challenges with testing, and it really depends on situations. This means we'd have to consult all business counterparts/ team members to get an idea. To note, our current setting only has records of 5 days after the new system went live; so the performance or any results are highly doubtful if it was meant to be a universal system change.

3.3. Difficulties (Less controllable)

As much prep work as we want to do for our testing/ experiment, there are only so many factors we can actually have greater control. Below are the things to watch out for when conducting analyses or drawing results:

- **Novelty Effect:** Depends on how customers could feel the change, data gathered might not be completely useful due to novelty effect. That is, certain 'good numbers' might just so happen because, for instance, the users find the new UI funny or they like the colors, but not because they find it become truly more user friendly. Chances are they might ditch the app after a few days of change.
- **Change Aversion:** Sometimes users just don't like to change or even try, for no reasons, and there is really nothing you can do about it. It's also worth noting that this is quite a common phenomenon among existing users who first experience a change.