TaskRabbit Data Exercise (Mingliang Zhou)

GitHub: https://github.com/MingliangZhou/DS_Project_TR

In [1]:

```
import numpy as np
import pandas as pd

import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline

from jupyterthemes import jtplot
jtplot.style(theme='grade3')

import warnings
warnings.filterwarnings('ignore')
```

Question 1

How many recommendation sets are in this data sample?

In [2]:

```
df = pd.read_csv('../input/sample.csv')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30000 entries, 0 to 29999
Data columns (total 8 columns):
recommendation_id      30000 non-null object
created_at             30000 non-null object
tasker_id              30000 non-null int64
position               30000 non-null int64
hourly_rate            30000 non-null int64
num_completed_tasks    30000 non-null int64
hired                  30000 non-null int64
category               30000 non-null object
dtypes: int64(5), object(3)
memory usage: 1.8+ MB
```

In [3]:

```
df['recommendation_id'].nunique()
```

Out[3]:

```
2100
```

2100 recommendation sets

Question 2

Each recommendation set shows from 1 to 15 Taskers, what is:

- average number of Taskers shown
- median number of Taskers shown

In [4]:

```
df.groupby('recommendation_id')['tasker_id'].nunique().agg(['mean', 'median'])
```

Out[4]:

```
mean      14.285714
median    15.000000
Name: tasker_id, dtype: float64
```

Average is 14.29
Median is 15

Question 3

How many total unique Taskers are there in this data sample?

In [5]:

```
df['tasker_id'].nunique()
```

Out[5]:

```
830
```

830 unique Taskers

Question 4

Which Tasker has been shown the most?
Which Tasker has been shown the least?

In [6]:

```
dfTaskerShown = df.groupby('tasker_id')['recommendation_id'].count().\
    reset_index(name='countTaskerShown').sort_values(by='countTaskerShown', ascending=False)
dfTaskerShown.head()
```

Out[6]:

	tasker_id	countTaskerShown
780	1014508755	608
502	1012043028	438
795	1014675294	387
789	1014629676	311
183	1008887321	290

In [7]:

```
dfTaskerShown[dfTaskerShown['countTaskerShown']==1]['tasker_id'].values
```

Out[7]:

```
array([1007383273, 1007246122, 1007295623, 1007480912, 1007638825,
       1006690425, 1014593279, 1006853970, 1006899551, 1010681878,
       1007472083, 1014926743, 1014547884, 1008919567, 1014478773,
       1010042971, 1009547227, 1011968845, 1009603880, 1012678504,
       1009612428, 1012386513, 1009618500, 1012364558, 1009641175,
       1011972750, 1012348656, 1012289475, 1010779242, 1009702351,
       1009712638, 1009754999, 1011985968, 1012166729, 1009772528,
       1012151299, 1010021990, 1010009736, 1012071620, 1009871933,
       1012805440, 1011957940, 1009461190, 1011952623, 1014439502,
       1014310300, 1007923586, 1008033678, 1014086818, 1013934937,
       1013854788, 1013830691, 1011901532, 1008368716, 1008469117,
       1013731883, 1008474216, 1013656032, 1008604368, 1008828652,
       1013573988, 1013573125, 1008870833, 1009994950, 1013362004,
       1009112003, 1011949117, 1010640007])
```

Tasker 1014508755 has been shown the most
More than 1 Tasker has been shown the least: [1007383273, 1007246122, 1007295623, 1007480912, 1007638825, 1006690425, 1014593279, 1006853970, 1006899551, 1010681878, 1007472083, 1014926743, 1014547884, 1008919567, 1014478773, 1010042971, 1009547227, 1011968845, 1009603880, 1012678504, 1009612428, 1012386513, 1009618500, 1012364558, 1009641175, 1011972750, 1012348656, 1012289475, 1010779242, 1009702351, 1009712638, 1009754999, 1011985968, 1012166729, 1009772528, 1012151299, 1010021990, 1010009736, 1012071620, 1009871933, 1012805440, 1011957940, 1009461190, 1011952623, 1014439502, 1014310300, 1007923586, 1008033678, 1014086818, 1013934937, 1013854788, 1013830691, 1011901532, 1008368716, 1008469117, 1013731883, 1008474216, 1013656032, 1008604368, 1008828652, 1013573988, 1013573125, 1008870833, 1009994950, 1013362004, 1009112003, 1011949117, 1010640007]

Question 5

Which Tasker has been hired the most?
Which Tasker has been hired the least?

In [8]:

```
dfTaskerHired = df[df['hired']==1].groupby('tasker_id')['recommendation_id'].count().\
    reset_index(name='countTaskerHired').sort_values(by='countTaskerHired', ascending=False)
dfTaskerHired.head()
```

Out[8]:

	tasker_id	countTaskerHired
195	1012043028	59
222	1013131759	39
236	1013359522	37
225	1013165984	36
261	1013794735	35

In [9]:

```
dfTaskerHired[dfTaskerHired['countTaskerHired']==1]['tasker_id'].values
```

Out[9]:

```
array([1008111352, 1007477780, 1006720321, 1015009096, 1014832736,
       1007164698, 1007146669, 1007898815, 1013852438, 1014740602,
       1013865107, 1007480912, 1008037901, 1014251534, 1014629676,
       1008008634, 1014100703, 1007702812, 1014478773, 1014157578,
       1013745898, 1007955495, 1014212647, 1008030151, 1008862713,
       1013745838, 1009966564, 1011920226, 1011914012, 1009299585,
       1010801075, 1010752503, 1009348455, 1009393887, 1010578972,
       1010565292, 1010438496, 1010008242, 1009606144, 1013711863,
       1009917428, 1009865854, 1009848016, 1009616142, 1009751605,
       1009733517, 1009729754, 1009703547, 1009638159, 1009693303,
       1009638573, 1011985968, 1009230070, 1009192067, 1012170634,
       1013696131, 1013677268, 1008162664, 1013553854, 1008473496,
       1013443125, 1013434046, 1008724560, 1013305557, 1008762938,
       1008782548, 1008790779, 1008887321, 1013034579, 1008890855,
       1012719266, 1012666042, 1008903001, 1008930827, 1012278216,
       1008962854, 1008966829, 1012229493, 1015020347])
```

Tasker 1012043028 has been hired the most
More than 1 Tasker has been hired the least (excluded Taskers who never been hired): [1008111352, 1007477780, 1006720321, 1015009096, 1014832736, 1007164698, 1007146669, 1007898815, 1013852438, 1014740602, 1013865107, 1007480912, 1008037901, 1014251534, 1014629676, 1008008634, 1014100703, 1007702812, 1014478773, 1014157578, 1013745898, 1007955495, 1014212647, 1008030151, 1008862713, 1013745838, 1009966564, 1011920226, 1011914012, 1009299585, 1010801075, 1010752503, 1009348455, 1009393887, 1010578972, 1010565292, 1010438496, 1010008242, 1009606144, 1013711863, 1009917428, 1009865854, 1009848016, 1009616142, 1009751605, 1009733517, 1009729754, 1009703547, 1009638159, 1009693303, 1009638573, 1011985968, 1009230070, 1009192067, 1012170634, 1013696131, 1013677268, 1008162664, 1013553854, 1008473496, 1013443125, 1013434046, 1008724560, 1013305557, 1008762938, 1008782548, 1008790779, 1008887321, 1013034579, 1008890855, 1012719266, 1012666042, 1008903001, 1008930827, 1012278216, 1008962854, 1008966829, 1012229493, 1015020347]

Question 6

If we define the "Tasker conversion rate" as the number of times a Tasker has been hired, out of the number of times the Tasker has been shown, how many Taskers have a conversion rate of 100%?

In [10]:

```
dfTaskerConv = pd.merge(dfTaskerShown, dfTaskerHired, on='tasker_id', how='left').fillna(0)
dfTaskerConv['tasker_conversion_rate'] = dfTaskerConv['countTaskerHired'] /\
    dfTaskerConv['countTaskerShown']
dfTaskerConv[dfTaskerConv['tasker_conversion_rate']==1]
```

Out[10]:

	tasker_id	countTaskerShown	countTaskerHired	tasker_conversion_rate
505	1008861741	9	9.0	1.0
734	1008094420	2	2.0	1.0
750	1012369686	2	2.0	1.0
765	1007480912	1	1.0	1.0
776	1014478773	1	1.0	1.0
794	1011985968	1	1.0	1.0

6 Taskers

Question 7

Would it be possible for all Taskers to have a conversion rate of 100%? Please explain your reasoning.

Since each recommendation hires only one Tasker, 100% conversion rate for all Taskers can only happen if each recommendation has only one Tasker shown, and that Tasker is also hired.

Question 8

For each category, what is the average position of the Tasker who is hired?

In [11]:

```
df[df['hired']==1].groupby('category')['position'].mean()
```

Out[11]:

```
category
Furniture Assembly    3.611888
Mounting              4.596085
Moving Help           4.145359
Name: position, dtype: float64
```

Furniture Assembly: 3.61
Mounting: 4.60
Moving Help: 4.15

Question 9

For each category, what is the average hourly rate and average number of completed tasks for the Taskers who are hired?

In [12]:

```
df[df['hired']==1].groupby('category')[['hourly_rate', 'num_completed_tasks']].mean()
```

Out[12]:

	hourly_rate	num_completed_tasks
category		
Furniture Assembly	38.701049	249.020979
Mounting	50.154804	284.096085
Moving Help	63.012259	273.882662

Average hourly rate
Furniture Assembly: 38.70
Mounting: 50.15
Moving Help: 63.01

Average number of completed tasks
Furniture Assembly: 249.02
Mounting: 284.10
Moving Help: 273.88

Question 10

Based on the previous, how would you approach the question of: How can we use market data to suggest hourly rates to Taskers that would maximize their opportunity to be hired? Please describe in detail, with code and formulas that support your model.

One of the confounding attributes that determine the opportunity to be hired is the service quality of a Tasker. This is why we observed some taskers are still hired even though their hourly rate is higher than their competitors.

Unfortunately, the attribute that reflects the service quality is not included in the data. num_completed_tasks is a poor estimate of service quality since it doesn't take running time into account, thus unfair to relatively new Taskers.

However, for each Tasker, service quality can be assumed to remains the same. We might get some insights here. The question is: do we have enough statistics and variation of hourly rates for each Tasker?

In [13]:

```
df.groupby(['category', 'tasker_id'])['hourly_rate'].agg(['count', 'nunique']).\
    reset_index().sort_values(by=['count'], ascending=False).head(10)
```

Out[13]:

	category	tasker_id	count	nunique
786	Mounting	1014508755	522	1
790	Mounting	1014629676	311	5
109	Furniture Assembly	1008887321	256	1
783	Mounting	1014423146	240	2
1171	Moving Help	1013283344	227	3
794	Mounting	1014675294	223	2
301	Furniture Assembly	1012043028	217	8
1242	Moving Help	1014157578	215	3
712	Mounting	1012721277	214	9
689	Mounting	1012043028	214	5

Unfortunately, there are neither enough statistics nor variation of hourly rates to extract useful information for each Tasker. This means we have to combine all Taskers together.

Even so, by assuming that the service qualities of the 'normal' Taskers (hourly rates within IQR) are roughly the same. We could might still spot some pattern.

Since each category have very different average/median hourly rates, the analysis is performed in each category. First, fill the histogram of hourly rates for each category, in three cases: hired, not hired and combined.

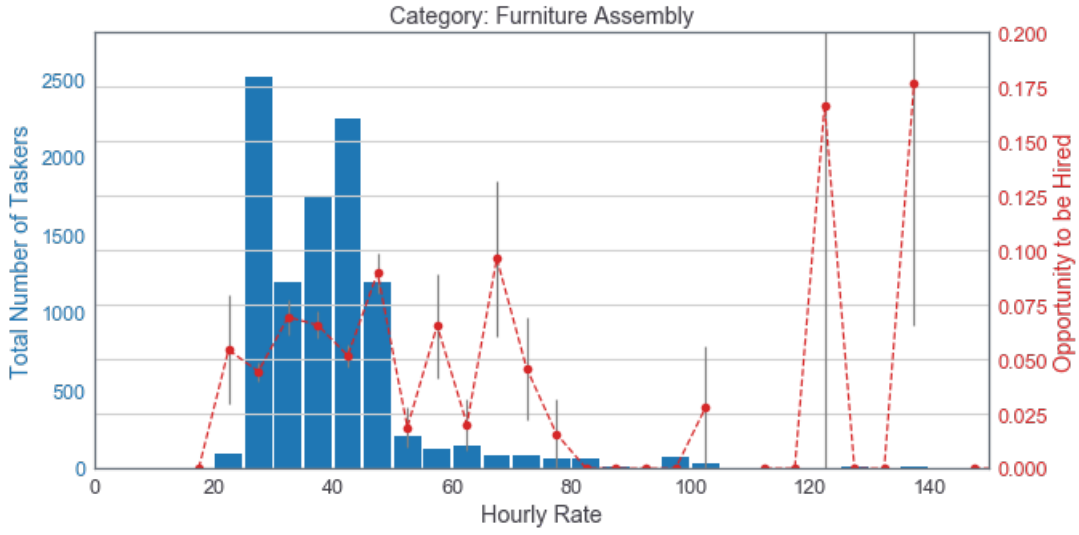
In [14]:

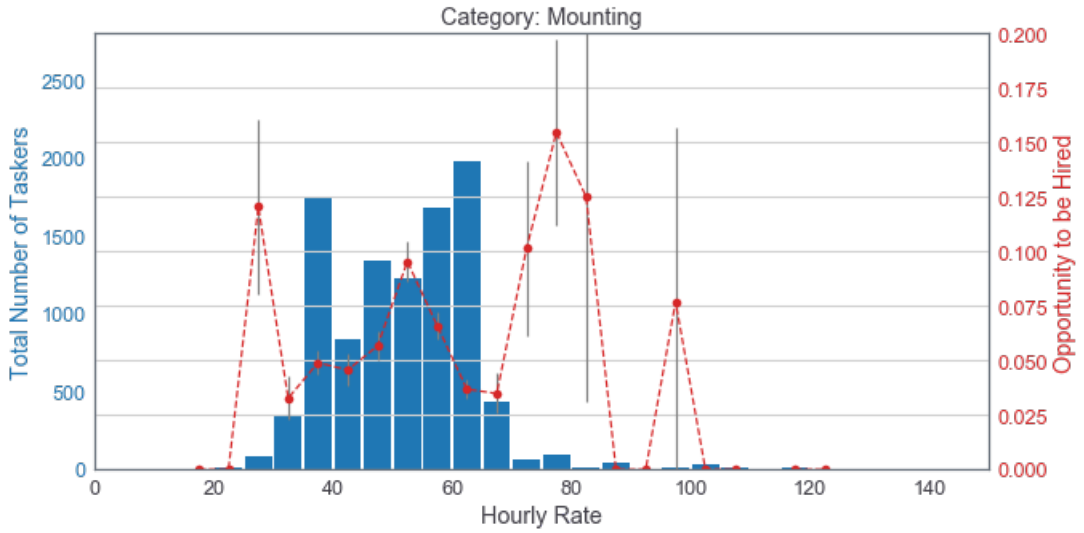
```
categories = ['Furniture Assembly', 'Mounting', 'Moving Help']
hist_hourly_rate = [] # 3 categories | hired, not hired, combined
for category in categories:
    hired = np.histogram(df[(df['hired']==1) & (df['category']==category)]['hourly_rate']\
            .values, bins=np.linspace(0,200,41))
    unhired = np.histogram(df[(df['hired']==0) & (df['category']==category)]['hourly_rate']\
            .values, bins=np.linspace(0,200,41))
    combined = np.histogram(df[df['category']==category]['hourly_rate']\
            .values, bins=np.linspace(0,200,41))
    hist_hourly_rate.append([hired, unhired, combined])
```

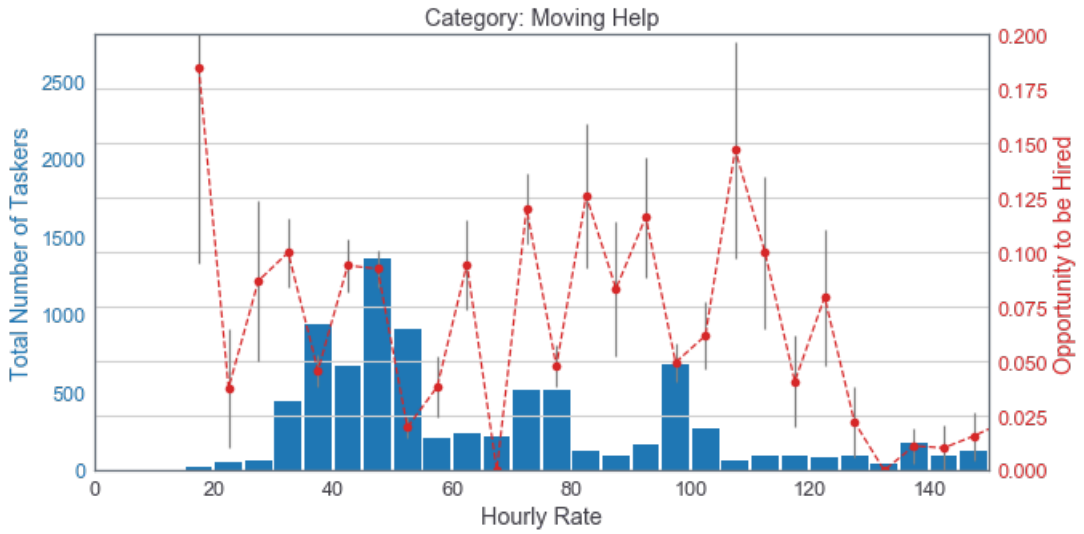
Within each hourly rate range, the opportunity to be hired is defined as the number of hired Taskers out of the total number of shown taskers. They are shown in the following plots.

In [15]:

```
xpos = np.linspace(2.5,197.5,40)
for iC in range(len(categories)):
    A, B = hist_hourly_rate[iC][0][0], hist_hourly_rate[iC][2][0]
    yerr = A/B*np.sqrt(1./A+1./B)
    fig = plt.figure(figsize=(10,5))
    ax1 = fig.add_subplot(1,1,1)
    ax1.grid(False)
    ax1.set_title('Category: {}'.format(categories[iC]))
    ax1.set_xlabel('Hourly Rate')
    ax1.set_ylabel('Total Number of Taskers', color='tab:blue')
    ax1.bar(xpos, hist_hourly_rate[iC][2][0], width=4.5, color='tab:blue')
    ax1.tick_params(axis='y', labelcolor='tab:blue')
    ax1.set_xlim([0,150])
    ax1.set_ylim([0,2800])
    ax2 = ax1.twinx()
    ax2.set_ylabel('Opportunity to be Hired', color='tab:red')
    ax2.errorbar(xpos, hist_hourly_rate[iC][0][0]/hist_hourly_rate[iC][2][0],
                 yerr=yerr, ecolor='gray', fmt='o--', color='tab:red')
    ax2.set_ylim([0,0.2])
    ax2.tick_params(axis='y', labelcolor='tab:red')
```







For furniture assembly, most Taskers have hourly rate between 25 to 50. The opportunity does not change dramatically as a function of hourly rate. Thus a higher rate is suggested in order to increase revenue: we suggest 45 to 50.

For mounting, most Taskers have hourly rate between 35 to 65. The opportunity is significantly higher for hourly rate between 50 to 55. Comparing hourly rate range 50-55 to 55-60, the expected revenue per order is increased by 0.0954 *52.5 - 0.0660* 57.5 = 1.213. This means that 50-55 hourly rate not only increases the opportunity to be hired, but also increases revenue overall. Thus we suggest 50-55.

For moving help, the situation is more complicated. There are multiple clusters of hourly rates: around 50, 70 and 100 respectively. This is expected since compared with furniture assembly and mounting, moving requires more expertises and the service quality will decide the opportunity to be hired. So more information is required and the suggestions will depend on different clusters of Taskers.

Suggested hourly rate for Taskers working on 'furniture assembly': 45 - 50
Suggested hourly rate for Taskers working on 'mounting': 50 - 55
Suggested hourly rate for Taskers working on 'furniture assembly': more info needed