Final_Project (2)

June 10, 2020

1 Ecommerce Analysis Final Project

Group Menbers: Shuyun Tang and Yuehan Guo

2 Part I Abstract(100+)

One paragraph that briefly summarizes the problem you addressed, questions of interest, the data, the techniques used, findings and key results, and conclusions.

Part 2 Introduction (200 words or more):**

In this section you should describe 1. the primary goal(s) or question(s) that your project addresses, 1. the motivation for your project, i.e., why your readers should be interested, 1. the relevant background of your topic, including a brief literature review (a paragraph with 1-3 references) describing any prior related work. 1. the dataset you are using to answer your question. You should address why this data is appropriate for answering your question.

3 Part II Introduction

Nowadays, online shopping is becoming more and more popular. People could shop whatever they want at home with a few finger clicks, and need not to consider all those bothersome transportations to get to the shopping mall and the energy and time spent with the in-store assistants. People could even spend more time on comparing different products and deciding what to buy. Driven by this trend, it is very important for eshop owners to explore what is really valued by customers in order to make more profit. Besides those eshop owners, other readers who are online shopper can gain insightful, experienced reports from most of the other online shoppers. So they can aviod some fraud info in the eshops.

The primary goal of our project is to explore differenct influencing factors (namely product price, delivery length, payment method, quantity of product photos, and the geographic position of customers) on customer's e-purshace bahaviors and satisfaction, so that we could predict customers' purchase preference and the trend of ecommerce.

Shuyun is an experienced accessories e-shop owner, and he would like to get some inspiration on how to improve his e-shop to make more benefit by analyzing the customer data; and Yuehan is an experienced online shopper who is very familiar with different factors involved in online shopping

that could potentially affect customers' behavior and preference. Hence, this data exploration is especially meaningful and interesting for us.

This data was from https://www.kaggle.com/jainaashish/orders-merged, and it doesn't have a specific license. The publisher does not provide detailed information either. However, after our analysis, we found out that most of the buyers were located in Brazil and that this was a real 100k+ realtime local online orders dataset in Brazil in 2019. It provides lots of clean, insightful data about those influencing factors listed above and thus, it is appropriate for us to use.

3.1 Questions of interest:

exploring the correlation of the following and provide the feedbacks: 1. Delivery length vs review score 2. Customer state & purchase behavior/ payment amount/ payment method 1. product photo quantity & purchase behavior 2. payment amount / payment method (what is the most popular paymentmethod)

[0]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import altair as alt
import os
from sklearn.linear_model import LinearRegression
# import altair as alt
%matplotlib inline
pd.set_option('display.max_columns', None)
```

[0]:
```
from datetime import datetime
```

[0]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id
=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redire
ct_uri=urn%3aietf%3awg%3aoauth%3a2.0%3aoob&response_type=code&scope=email%20http
s%3a%2f%2fwww.googleapis.com%2fauth%2fdocs.test%20https%3a%2f%2fwww.googleapis.c
om%2fauth%2fdrive%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive.photos.reado
nly%20https%3a%2f%2fwww.googleapis.com%2fauth%2fpeopleapi.readonly

Enter your authorization code:
··········
Mounted at /content/drive
```

4 Ecommerce Data

This data was from https://www.kaggle.com/jainaashish/orders-merged, and the goal of this dataset is to see what factors influence customers' online shopping behaviors. *ADD MORE ABOUT THE DATASET

[0]:
```
customers = pd.read_csv("/content/Orders_merged.csv")
```

[0]:
```
customers.head()
```

[0]:
```
                          product_id  … product_width_cm
0  00066f42aeeb9f3007548bb9d3f33c38  …              16.0
1  00088930e925c41fd95ebfe695fd2655  …              26.0
2  0009406fd7479715e4bef61dd91f2462  …              35.0
3  000b8f95fcb9e0096488278317764d19  …              12.0
4  000b8f95fcb9e0096488278317764d19  …              12.0

[5 rows x 38 columns]
```

5 Data Cleaning

Since there are too many columns, we should create a new dataframe with only the columns that we are interested in. Let's create a dataframe **"df"**, which is the target column **'review_score'**(customers' satisfaction) with the "influencing factors" colums **'order_purchase_timestamp'**(when did the customers place the orders), **'order_estimated_delivery_date'**(when did the customers estimated to receive), **'product_description_lenght'**(the description length listed in the e-shops) **'product_photos_qty'**(how many photos did the shopper provide), **'price'**, **'freight value'** (the shipping fee). Those columns are enough for our questions of interest.

[105]:
```
# make a new df contained the above columns
df=customers.drop(columns=["order_status","order_delivered_carrier_date",
           "order_approved_at","customer_zip_code_prefix","customer_unique_id",
          ␣
 ↪"customer_city","review_id","review_comment_title","review_comment_message",
          ␣
 ↪"review_creation_date","review_answer_timestamp","order_item_id","seller_zip_code_prefix",
           "seller_city","product_category_name","product_id","seller_id",
           "order_id","customer_id",
          ␣
 ↪"customer_state","seller_state","payment_sequential","payment_value","payment_type","produc
          ␣
 ↪"product_height_cm","product_length_cm","product_width_cm","payment_installments","product_

# we will drop the null values here
df=df.dropna()
```

```
df.head()
```

[105]:
```
  order_purchase_timestamp order_delivered_customer_date  \
0      2018-05-20 18:45:00           2018-06-06 22:11:00
1      2017-12-12 19:20:00           2017-12-23 17:11:00
2      2017-12-21 16:21:00           2018-01-06 15:03:00
3      2018-08-01 22:00:00           2018-08-07 17:38:00
4      2018-08-10 13:24:00           2018-08-17 21:33:00

  order_estimated_delivery_date  review_score   price  freight_value  \
0           2018-06-20 00:00:00             5  101.65          18.59
1           2018-01-05 00:00:00             4  129.90          13.93
2           2018-01-16 00:00:00             1  229.00          13.10
3           2018-08-24 00:00:00             5   58.90          19.60
4           2018-08-27 00:00:00             5   58.90          19.60

   product_description_lenght  product_photos_qty
0                       596.0                6.0
1                       752.0                4.0
2                       266.0                2.0
3                       364.0                3.0
4                       364.0                3.0
```

5.1 Features Engineering

We will then add some new features based on the current columns: **del_time** and **est_del_time** are the datetime conversion of the 'order_delivered_customer_date' and 'order_estimated_delivery_date' columns. **timediff** is the delivery time minus the estimated delivery time and converts to the days difference to the integers. **diliver_time** is the delivery time minus the purchase time and converts to the integers. **purchase_weekday** is the weekday of the purchase time, in which 0 is Monday and 6 is Sunday. **Late** is the categorical variable that indicates 0 if the timediff is less than 0, which means delivery time is earlier than estimated delivery time, or the delivery is not late than the estimated time and vice versa.

We will add those columns to our dataframe and explore their relationships to the review score.

[106]:
```
df['del_time']= pd.to_datetime(df['order_delivered_customer_date']).dt.
 ↪floor('d')
df['est_del_time']= pd.to_datetime(df['order_estimated_delivery_date'])

df['order_purchase_timestamp'] = pd.to_datetime(df['order_purchase_timestamp']).
 ↪dt.floor('d')
df['timediff']=df['del_time']-df['est_del_time']
df['timediff'] = pd.to_numeric(df['timediff'].dt.days, downcast='integer')

df['deliver_time']=df['del_time']-df['order_purchase_timestamp']
```

```
df['deliver_time'] = pd.to_numeric(df['deliver_time'].dt.days,␣
↪downcast='integer')

df['purchase_weekday']=df['order_purchase_timestamp'].dt.weekday ##Monday=0,␣
↪Sunday=6.

df['Late']=np.where(df['timediff']<0,0,1) ##0 means timediff<0, actual del time␣
↪is earlier than estimated del
df=df.
↪drop(columns=['order_delivered_customer_date','order_estimated_delivery_date','del_time','o

df.head()
```

[106]:

	review_score	price	freight_value	product_description_lenght	\
0	5	101.65	18.59	596.0	
1	4	129.90	13.93	752.0	
2	1	229.00	13.10	266.0	
3	5	58.90	19.60	364.0	
4	5	58.90	19.60	364.0	

	product_photos_qty	timediff	deliver_time	purchase_weekday	Late
0	6.0	-14	17	6	0
1	4.0	-13	11	1	0
2	2.0	-10	16	3	0
3	3.0	-17	6	2	0
4	3.0	-10	7	4	0

5.2 Review Score Category redesign

As the eshop owner, Shuyun will usually classify the 1, 2 review scores as the "bad reviews".

In order to make our model more accurate, we will sort out the review scores lower or equal than 2 into the negative reviews, which indicated as 0; and the review scores higher than 2 into the positive reviews, which indicated as 1.

[0]:
```
df['review_score']=(df['review_score']>2)+0
```

6 Pre-analysis Data Exploring

Before we can begin with data analysis, we need to do a little bit of data exploration.

- Are there any missing values? If so, how many?

[0]:
```
customers.isnull().sum().sum()
#total count is 3522735, rows are 96479
```

[0]:
```
147613
```

[0]:
```
df.isnull().sum().sum()
# df after adjusting columns total Nah count is 0
```

[0]:
```
0
```

- Are there any unique values? Any values that seems strange?

All the values are in our expectation and no unique values. Some large payment values are probably luxury goods.

[0]:
```
df.info
```

[0]:
```
<bound method DataFrame.info of        review_score   price  …
purchase_weekday  Late
0                 5  101.65  …                 6     0
1                 4  129.90  …                 1     0
2                 1  229.00  …                 3     0
3                 5   58.90  …                 2     0
4                 5   58.90  …                 4     0
…               …     …  …               …   …
96473             5   34.99  …                 1     0
96474             5   29.99  …                 6     1
96475             5   34.99  …                 4     0
96476             5   34.99  …                 3     0
96477             4  249.99  …                 2     1

[95114 rows x 9 columns]>
```

[0]:
```
df.describe() #all values are making sense
```

[0]:
```
       review_score         price  …  purchase_weekday          Late
count  95114.000000  95114.000000  …      95114.000000  95114.000000
mean       4.144374    125.308130  …          2.755861      0.080987
std        1.294361    189.635046  …          1.966987      0.272817
min        1.000000      0.850000  …          0.000000      0.000000
25%        4.000000     41.792500  …          1.000000      0.000000
50%        5.000000     79.000000  …          3.000000      0.000000
75%        5.000000    139.900000  …          4.000000      0.000000
max        5.000000   6735.000000  …          6.000000      1.000000

[8 rows x 9 columns]
```

- Which parts of the data were entered by a human? Are there any other potential sources of error?

  All the data is supposed to be generated by the e-shop platform, and there should be no data entered by a human after our obeservation to the data source. There might be

some potential sources of error on the other columns such as the length, weight of the products. But our dataframe will not address those potential columns here.

Also, there are some noticable large or small delivery time. The reasons are probably custom regulation or the pre-order sales.

- What are the ethical considerations regarding this dataset?

  The primary ethical considerations are the privacy of each customers, especially their payment infomation. Thus, when generating the processed dataset, we will exclude those sensitive columns to protect their infomation.

  Another consideration is their locations. Since the original dataset contains their exact address and zip code, it is not safe to use them in our final project without the instructions of the original pubulisher. So we exclude those columns in our processed dataset.

  There is no specific groups in our processed dataset over-represented or underrepresented. But the locations are in Brazil, there might lack some diversity due to the cultural difference.

- What principles of measurement" might be particularly relevant to your questions (distortion, relevance, precision, cost).

  The relevance is our main focus to those four questions as we are exploring their correlation and tendency. Then we can interprete the relevance and give the feedback.

6.1 EDA

6.1.1 Our ultimate questions are how to make the customers more satisfied, and other factors that can give us a better understanding in marketing. To be specific, we will explore the following:

1. Delivery length& delivery late vs review score
2. Freight value & purchase behavior& payment amount vs review score
3. product photo quantity&description length vs review score and price
4. Beyond the processed features, we will also analyze some interesting patterns from the original dataset–**bonus parts**: Which weekday usually has the most buyers? Which region usually has the most buyers and why? Which payment methods is the most popular? What other insightful thoughts will be drilled from those patterns?

[0]:
```
#disable the Altair maximum rows limit
alt.data_transformers.disable_max_rows()
```

[0]:
```
DataTransformerRegistry.enable('default')
```

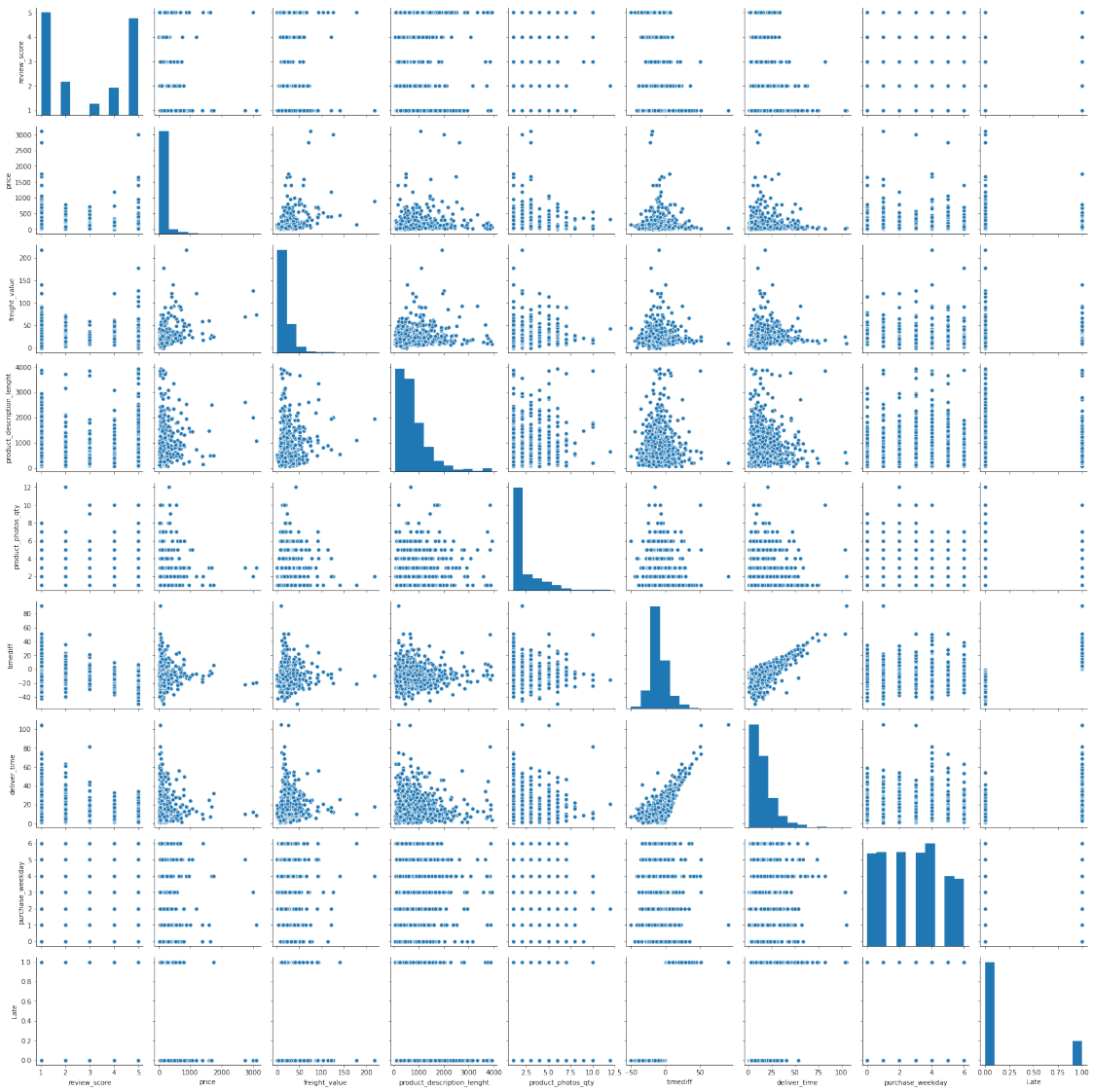
##Pair plot Let's create a pair-plot first to have a brief glance of some of the potentially interesting relationships.

There are some noticable patterns between the **review score and delivery time**, **product description length, product photo quantity and price**, **delivery time and freight value**,

etc

[0]:
```
sns.pairplot(df)
```

[0]: <seaborn.axisgrid.PairGrid at 0x7fbae6af3ef0>



##Correlation Matrix We will then conduct the correlation matrix to further explore those relationships. The relationships are shown as: negative relationship between the timediff, delivery time, and late delivery vs the review score. It is obvious that most of the buyers don't wanna wait too long. Also, the freight value, price, product description length and product photo quantities all have the positive relationship. We can assumn the goods that are more complex and valuable usually have more detailed information listed in the eshops.

[135]:
```
# Compute the correlation matrix
corr = df.corr()

# Generate a custom diverging colormap
cmap = sns.diverging_palette(220, 10, as_cmap=True)

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr, cmap=cmap, vmax=.3, center=0,
            square=True, linewidths=.6, cbar_kws={"shrink": .6}).
↪set_title('correlation matrix')
```

[135]:
```
Text(0.5, 1.0, 'correlation matrix')
```

The delivery time clearly has a negative relationship to the review score. The review score 0 (negative review score) has more density in the longer delivery time

[136]:
```
alt.Chart(df).transform_density(
    'deliver_time',
    as_=['deliver_time', 'density'],
    groupby=['review_score']
).mark_area(orient='horizontal').encode(
    y='deliver_time',
    color='review_score:N',
    x=alt.X(
        'density:Q',
        stack='center',
        impute=None,
        title=None,
        axis=alt.Axis(labels=False, values=[0],grid=False, ticks=True),
    ),
    column=alt.Column(
        'review_score:N',
        header=alt.Header(
            titleOrient='bottom',
            labelOrient='bottom',
            labelPadding=0,
        ),
    )
).properties(
    title='Review Score vs Delivery Time'
).properties(
    width=100
).configure_facet(
    spacing=0
).configure_view(
    stroke=None
)
```

[136]:
```
alt.Chart(…)
```

##Photo Quantities vs Review Scores From the chart below we can discover that the review score 0 usually have a large density in the small photo quantities. The reason behind is probably the customers are misled by the fraud or imcomplete product photos and info.

[137]:
```
alt.Chart(df).transform_density(
    'product_photos_qty',
    as_=['product_photos_qty', 'density'],
    groupby=['review_score']
).mark_area(orient='horizontal').encode(
    y='product_photos_qty',
    color='review_score:N',
    x=alt.X(
        'density:Q',
```

```
        stack='center',
        impute=None,
        title=None,
        axis=alt.Axis(labels=False, values=[0],grid=False, ticks=True),
    ),
    column=alt.Column(
        'review_score:N',
        header=alt.Header(
            titleOrient='bottom',
            labelOrient='bottom',
            labelPadding=0,
        ),
    )
).properties(
    title='Review Score vs Photo Quantities'
).properties(
    width=100
).configure_facet(
    spacing=0
).configure_view(
    stroke=None
)
```

[137]:
```
alt.Chart(…)
```

##Payment values There exists a vague relationship between the payment values and the review score. The possible reason is that with a lower payment, the customers usually buy the cheap goods that have worse quality.

[138]:
```
alt.Chart(df).transform_density(
    'price',
    as_=['price', 'density'],
    groupby=['review_score']
).mark_area(orient='horizontal').encode(
    y='price',
    color='review_score:N',
    x=alt.X(
        'density:Q',
        stack='center',
        impute=None,
        title=None,
        axis=alt.Axis(labels=False, values=[0],grid=False, ticks=True),
    ),
    column=alt.Column(
        'review_score:N',
        header=alt.Header(
            titleOrient='bottom',
```

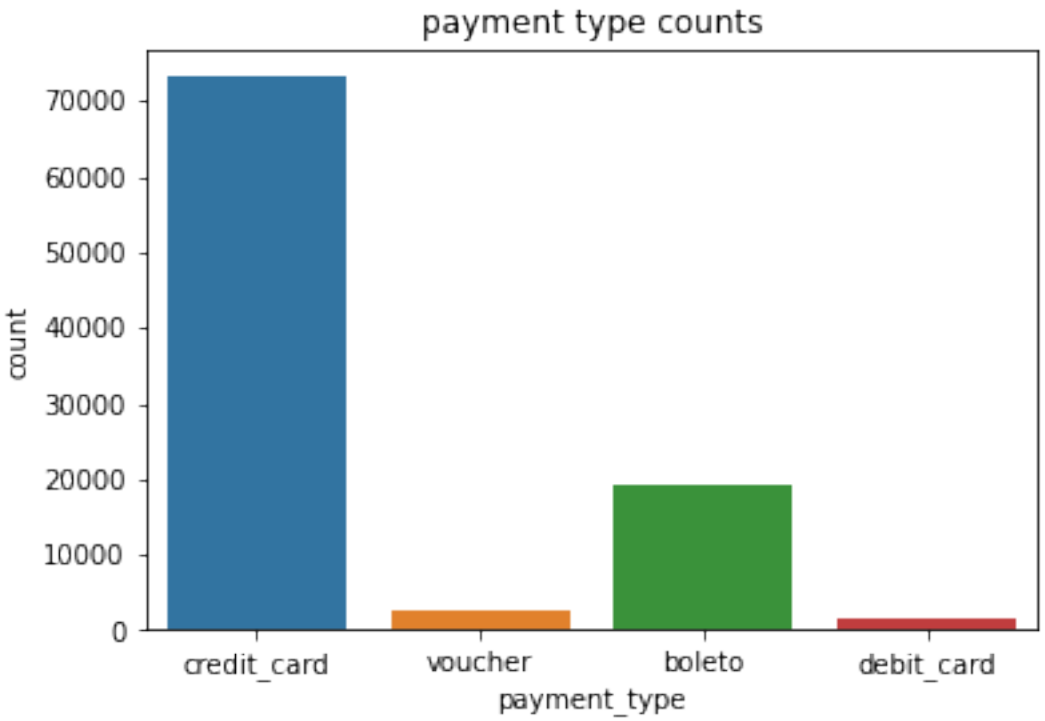
```
            labelOrient='bottom',
            labelPadding=0,
        ),
    )
).properties(
    title='Review Score vs Price'
).properties(
    width=100
).configure_facet(
    spacing=0
).configure_view(
    stroke=None
)
```

[138]: `alt.Chart(…)`

##Payment type (Bonus) from this plot we can find out the most frequent payment type is credit card. The second one is boleto, which is a growing payment method popular in Latin America. The debit card and voucher are very rare to be used.

[0]:
```
sns.countplot(x='payment_type',data=customers).set_title('payment type counts')
```

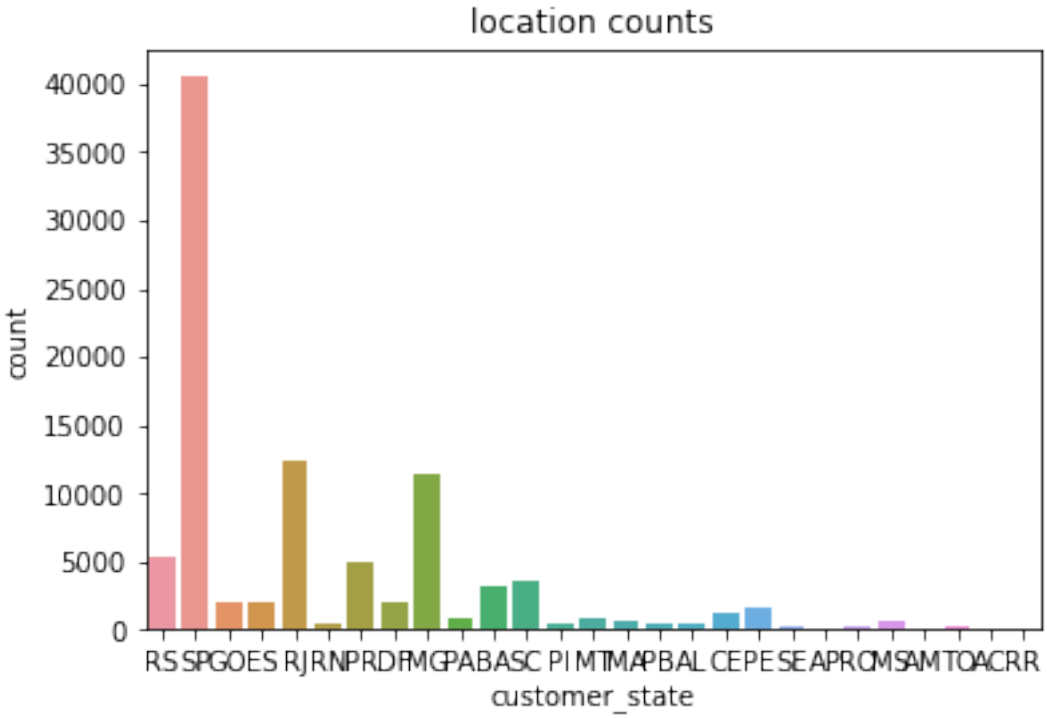
[0]: `Text(0.5, 1.0, 'payment type counts')`



##Location (Bonus) From the plot below we can discover that the SP (São Paulo) has the most customers. Because São Paulo is the municipality with large population (world's 12th largest city proper by population). Other two state RJ (Rio, lots of carnivals), and MG (a large inland colonial state before, prosperity phase of economy)

[0]:
```
sns.countplot(x='customer_state',data=customer_cleaned).set_title('location␣
↪counts')
```

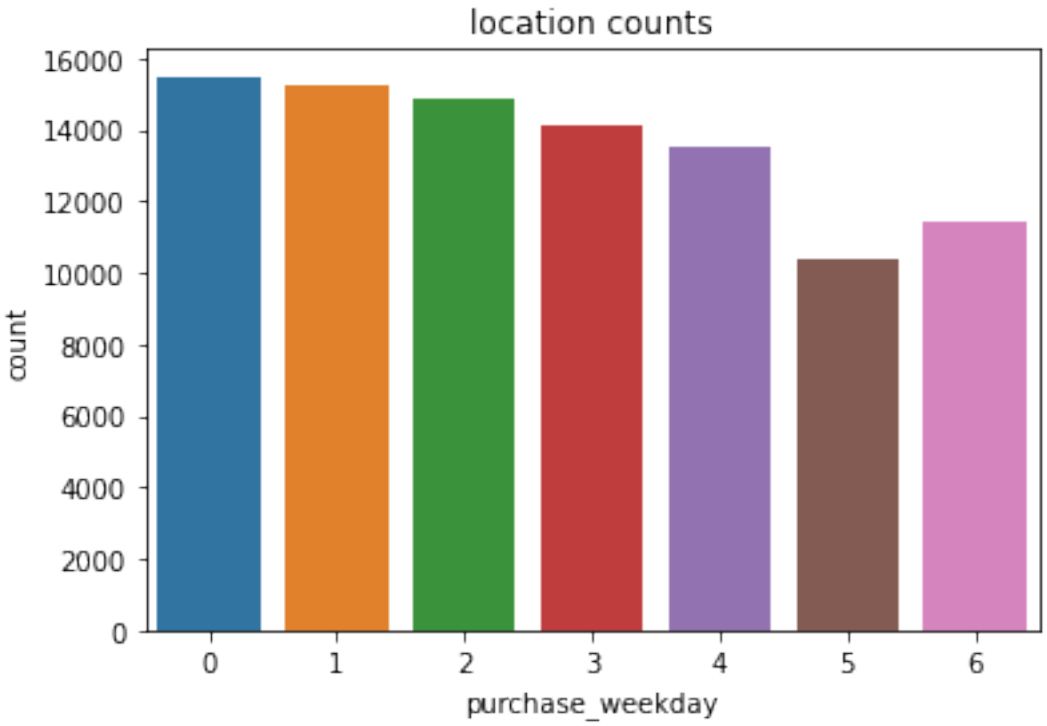
[0]:
```
Text(0.5, 1.0, 'location counts')
```



##Weekdays (Bonus) The Monday has the most customers. Based on our experience, it is not hard to deduce that people like to shopping at the beginning of the week and expect to recieve them in the weekends.

[0]:
```
sns.countplot(x='purchase_weekday',data=df).set_title('weekdays counts')
```

[0]:
```
Text(0.5, 1.0, 'location counts')
```



7 Sampling, partitioning and balancing the data

Due to the extremely large sample, which is more than 90000 rows, it will take lots of time and memory to analyze all of them. Also, the positive reviews are significantly more than the negative reviews (people are nice). We will sample and partition the dataset into the classfication columns and target columns before we put it into our model.

##sampling the balanced dataset we will take 2000 random samples. 1000 from those review score higher than 2 (positive reviews) and 1000 from those lower or equal than 2 (negative reviews) So that it will make the sample set more balanced and easy to analyze.

[109]:
```
import random
n_sample = 2000
df = df.dropna()
df1 = df[df['review_score']==1]
df2 = df[df['review_score']==0]
df1 = df1.sample(n_sample)
df2 = df2.sample(n_sample)
df_balanced = pd.concat([df1,df2],axis=0)
df_target_balanced=df_balanced['review_score']
df_classfication_balanced=df_balanced.drop(columns=['review_score'])
##df_classfication
```

```
df_classfication_balanced
```

[109]:
```
        price  freight_value  product_description_lenght  product_photos_qty  \
2723    45.90          10.96                       473.0                 1.0
65281   69.90          12.43                       903.0                 6.0
72607   20.70          14.44                       598.0                 1.0
22246  262.50         206.45                       387.0                 1.0
64011  174.90          30.26                       289.0                 1.0
…         …              …                           …                   …
38871  359.00          17.95                       537.0                 5.0
30328  179.90          27.34                       269.0                 1.0
70443  120.90          74.58                       707.0                 5.0
91032  200.00          15.01                       162.0                 5.0
30053  499.99          19.94                       644.0                 1.0

       timediff  deliver_time  purchase_weekday  Late
2723        -13            18                 4     0
65281        -4             7                 0     0
72607        -8             8                 1     0
22246       -15            18                 4     0
64011        -8            41                 0     0
…          …             …                 …   …
38871       -12            14                 5     0
30328        20            43                 1     1
70443       -21             8                 3     0
91032        -9             6                 3     0
30053        14            47                 4     1

[4000 rows x 8 columns]
```

##sampling the unbalanced (original ratio) dataset The reason we do this is to compare these two datasets in the machine learning part. Both will have different outcomes and we will analyze the advantages of choosing the balanced dataset or not.

[110]:
```
n_sample = 4000
df_unbalanced = df.sample(n_sample)
df_target_unbalanced=df_unbalanced['review_score']
df_classfication_unbalanced=df_unbalanced.drop(columns=['review_score'])
df_classfication_unbalanced
```

[110]:
```
        price  freight_value  product_description_lenght  product_photos_qty  \
66155   42.00          11.73                       595.0                 1.0
58847   63.90          11.73                       785.0                 1.0
75219  169.90          18.48                       856.0                 1.0
6522   139.90          17.42                      1194.0                 3.0
35511  238.00          16.52                      2066.0                 1.0
…         …              …                           …                   …
```

```
92626  759.99          21.08                       536.0                 3.0
38590   68.70          16.24                      1031.0                 6.0
22722  299.99          18.35                       819.0                 4.0
66433   48.90          16.11                      1681.0                 2.0
51697   43.90          22.85                        84.0                 2.0

       timediff  deliver_time  purchase_weekday  Late
66155       -24             2                 2     0
58847        -9             6                 3     0
75219       -10            11                 0     0
6522        -13            11                 4     0
35511       -19             9                 1     0
…           …              …                 …     …
92626       -16            10                 4     0
38590       -30             8                 6     0
22722       -15             8                 6     0
66433       -10            12                 3     0
51697       -28            16                 0     0

[4000 rows x 8 columns]
```

7.1 Standardization

Standardize the classfication dataframe so that all the columns will have the same weight during the machine learning process.

[0]:
```
#function that standardize the input dataframe
def standardization(data):
    mu = np.mean(data, axis=0)
    sigma = np.std(data, axis=0)
    return (data - mu) / sigma
```

[0]:
```
df_classfication_balanced = standardization(df_classfication_balanced)
df_classfication_unbalanced = standardization(df_classfication_unbalanced)
```

8 Examine the four aspects with the help of Machine Learning (Sklearn)

The reason why we use logistic regression is that the review score is categorical variables rather than continues numerical variables. Thus, we need to apply the logistic regression to examine how well those classification columns (df_classification) could predict the target column (review score). The packages are used from sklearn.

##logistic regression about the balanced sample From the results above, the accuracy of our logistic model is very low. We observed the possible reasons:

- There are lots of factors such as the weekdays that are completely uncorrelated to the review score.
- The target column itself is hard to predict. Since the review scores are too subjective. And there are lots of people never leave their review scores (the systems will automatically assign the score 5).

[0]:
```
from sklearn.linear_model import LogisticRegression
logmodel = LogisticRegression()
logmodel2 = LogisticRegression()
```

[113]:
```
#do the train test split to our standerdized, balanced classification dataset
from sklearn.model_selection import train_test_split
X = df_classfication_balanced
y = df_target_balanced

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33,␣
↪random_state=42)
X_train.head(4)
```

[113]:

	price	freight_value	product_description_lenght	product_photos_qty	timediff	deliver_time	purchase_weekday	Late
9856	-0.279754	-0.164581	-0.579570	0.997965	-0.044842	0.429145	-0.899975	-0.463753
19133	-0.294648	-0.556345	-0.736627	-0.689687	0.262165	-0.348538	-1.402754	-0.463753
75876	-0.099766	-0.238111	0.719585	-0.689687	-1.042616	-0.504074	1.613921	-0.463753
36483	-0.458369	-0.667522	-0.288327	-0.127136	0.262165	-0.659611	-0.397196	-0.463753

[0]:
```
logmodel.fit(X_train,y_train)
pred = logmodel.predict(X_test)
```

[115]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       0.75      0.39      0.51       632
           1       0.61      0.88      0.72       688

    accuracy                           0.64      1320
   macro avg       0.68      0.63      0.62      1320
weighted avg       0.68      0.64      0.62      1320
```

##logistic regression about the unbalanced sample The accuracy for the unbalanced data is 0.88

(weighted avg is 0.88). It is significantly higher than the previous balanced sample. Why would this happen?

- The original ratio (a higher proportion of those give positive review scores) is a better fit to the model.
- The balanced data amplifies the incertainty of those give bad review scores.
- We assumn the model fits the positive review scores better. Because the pattens and relations in the good review scores are easy to find.

[130]:
```
X2 = df_classfication_unbalanced
y2 = df_target_unbalanced
X_train2, X_test2, y_train2, y_test2 = train_test_split(X2, y2, test_size=0.33,␣
↪random_state=42)
X_train.head(4)
```

[130]:

	price	freight_value	product_description_lenght	product_photos_qty	timediff	deliver_time	purchase_weekday	Late
9856	-0.279754	-0.164581	-0.579570	0.997965	-0.044842	0.429145	-0.899975	-0.463753
19133	-0.294648	-0.556345	-0.736627	-0.689687	0.262165	-0.348538	-1.402754	-0.463753
75876	-0.099766	-0.238111	0.719585	-0.689687	-1.042616	-0.504074	1.613921	-0.463753
36483	-0.458369	-0.667522	-0.288327	-0.127136	0.262165	-0.659611	-0.397196	-0.463753

[131]:
```
logmodel2.fit(X_train2,y_train2)
```

[131]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

[0]:
```
pred2 = logmodel2.predict(X_test2)
```

[133]:
```
print(classification_report(y_test2,pred2))
```

```
              precision    recall  f1-score   support

           0       0.55      0.35      0.43       153
           1       0.92      0.96      0.94      1167

    accuracy                           0.89      1320
   macro avg       0.74      0.66      0.69      1320
weighted avg       0.88      0.89      0.88      1320
```

##Random Forest about the balanced sample We will then conduct the random forest model. In short, it is a model that consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction

The reason we use it:

- Handles non-linear parameters efficiently: Non linear parameters don't affect the performance of a Random Forest unlike curve based algorithms. So, if there is high non-linearity between the independent variables, Random Forest may outperform as compared to other curve based algorithms.
- Random Forest is usually robust to outliers and can handle them automatically. And it will reduce the overfitting that existing in the decision tree model.

[0]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier

clf = AdaBoostClassifier()
clf.fit(X_train, y_train)
pred3 = clf.predict(X_test)

rfc = RandomForestClassifier()
rfc.fit(X_train, y_train)
pred4 = rfc.predict(X_test)
```

[127]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test,pred2))
```

```
              precision    recall  f1-score   support

           0       0.66      0.44      0.53       632
           1       0.61      0.79      0.69       688

    accuracy                           0.62      1320
   macro avg       0.63      0.62      0.61      1320
weighted avg       0.63      0.62      0.61      1320
```

[128]:
```
print(classification_report(y_test,pred3))
```

```
              precision    recall  f1-score   support

           0       0.63      0.52      0.57       632
           1       0.62      0.72      0.67       688

    accuracy                           0.63      1320
   macro avg       0.63      0.62      0.62      1320
weighted avg       0.63      0.63      0.62      1320
```

The result is still bad. We will then do the unbalanced sample for the random forest

##Random Forest about the unbalanced sample

[134]:
```
rfc2 = RandomForestClassifier()
rfc2.fit(X_train2, y_train2)
pred4 = rfc2.predict(X_test2)
print(classification_report(y_test2,pred4))
```

```
              precision    recall  f1-score   support

           0       0.65      0.30      0.41       153
           1       0.91      0.98      0.95      1167

    accuracy                           0.90      1320
   macro avg       0.78      0.64      0.68      1320
weighted avg       0.88      0.90      0.88      1320
```

0.90 accuracy is a satisfied level to us. Also, we can notice that the random forest would effectively reduce the outliers' impacts to the model. (And reduce the variance)

Will the unbalanced sample dataset be a proper fit into this model?

9 Discussion and Future Works