Coverage for **analysis_engine.py** : 90%

117 statements | 105 run | 12 missing | 0 excluded

```
1  import numpy as np
2  import pytest
3  import altair as alt
4  import pandas as pd
5  import datetime as dt
6
7  def generate_base_counts(data,metrics,grouped,top,asc=False,subset=(),choice = ()):
8      if subset is not ():
9          data_up = data.loc[data[subset]==choice]
10         data_up = data_up.groupby([grouped]).count().loc[:,[metrics]]
11     else:
12         data_up = data.groupby([grouped]).count().loc[:,[metrics]]
13     data_up.reset_index(inplace = True)
14     data_up = data_up.sort_values(by=metrics, axis = 0,ascending = asc)
15
16     return data_up.iloc[:top,:]
17
18
19 def generate_base_sum(data,metrics,grouped,top,asc=False,subset=(),choice = ()):
20     if subset is not ():
21         data_up = data.loc[data[subset]==choice]
22         data_up = data_up.groupby([grouped]).sum().loc[:,[metrics]]
23     else:
24         data_up = data.groupby([grouped]).sum().loc[:,[metrics]]
25     data_up.reset_index(inplace = True)
26     data_up = data_up.sort_values(by=metrics, axis = 0,ascending = asc)
27
28     return data_up.iloc[:top,:]
29
30
31 def conversions(data,grouped,metric,comp,subset=(),choice = ()):
32     if subset is not ():
33         nov_grouped = data.loc[data[subset]==choice]
34         nov_grouped = nov_grouped.groupby([grouped,'event_type']).count().loc[:,[metric]]
35     else:
36         nov_grouped = data.groupby([grouped,'event_type']).count().loc[:,[metric]]
37
38     nov_grouped[grouped] = nov_grouped .index.get_level_values(0)
39     nov_grouped['event_type'] = nov_grouped .index.get_level_values(1)
40     nov_grouped=nov_grouped.pivot(index=grouped, columns='event_type', values=metric).reset_index()
41     nov_grouped['v2c: views to cart'] = (nov_grouped.cart/nov_grouped.view)
42     nov_grouped['c2p: cart to payment'] = (nov_grouped.purchase/nov_grouped.cart)
43     nov_grouped['v2p: views to payment'] = (nov_grouped.purchase/nov_grouped.view)
44     nov_grouped = nov_grouped.loc[nov_grouped[grouped].isin(list(comp[grouped].unique()))]
45     nov_grouped_t = nov_grouped.T.loc[[grouped,'v2c: views to cart','c2p: cart to payment','v2p: views to payment']]
46     nov_grouped_t.columns = nov_grouped_t.loc[grouped,:]
47     nov_grouped_t = nov_grouped_t.drop(grouped)
48     nov_grouped_t.reset_index(inplace=True)
49     nov_grouped_t=nov_grouped_t.melt(id_vars=['event_type'],
50             var_name=grouped,
51             value_name="conversion_value")
52     return nov_grouped_t
53
54 def funnel(data,grouped,metric,subset=(),choice = ()):
55         if subset is not ():
56             nov_grouped = data.loc[data[subset]==choice]
57             nov_grouped = nov_grouped.groupby([grouped,'event_type']).count().loc[:,[metric]]
58         else:
59             nov_grouped = data.groupby([grouped,'event_type']).count().loc[:,[metric]]
60
61         nov_grouped[grouped] = nov_grouped .index.get_level_values(0)
```

```
        nov_grouped['event_type'] = nov_grouped .index.get_level_values(1)
        nov_grouped=nov_grouped.pivot(index=grouped, columns='event_type', values=metric).reset_index()
        nov_grouped = nov_grouped.dropna().T
        nov_grouped.columns = nov_grouped.loc[grouped,:]
        nov_grouped = nov_grouped.drop(grouped)
        nov_grouped.reset_index(inplace=True)
        nov_grouped =nov_grouped.melt(id_vars=['event_type'],
                var_name=grouped,
                value_name="funnel_value")
        return nov_grouped



nov = pd.read_csv('main_TopTranNv.csv')
nov.event_time = pd.to_datetime(nov["event_time"]).dt.date
sales_nov = nov.loc[nov.event_type=='purchase']
carts_nov = nov.loc[nov.event_type=='cart']
views_nov = nov.loc[nov.event_type=='view']


top_10_cat_sales = generate_base_counts(sales_nov,metrics = 'product_id',grouped = 'category_code',top = 10)
nov_funnel = funnel(nov,grouped = 'category_code',metric = 'user_session')
# print(nov_funnel)

top_10_cat_rev = generate_base_sum(sales_nov,metrics = 'price',grouped = 'category_code',top = 10)
nov_conversions = conversions(nov,grouped = 'category_code',metric = 'user_session',comp = top_10_cat_sales )
# print(top_10_cat_rev)


def plot_top_cat(top_10_cat_sales,top_10_cat_rev):
    c1 = alt.Chart(top_10_cat_sales,title = 'Top 10 categories by # of Sales').mark_bar().encode(
    x=alt.X('product_id:Q',title = '# of sales'),
    y = alt.Y('category_code:N',sort = '-x') ,
)
    c2 = alt.Chart(top_10_cat_rev,title = 'Total Revenue').mark_text().encode(
    y=alt.Y('category_code:N',axis = None,sort = '-text'),
    text='price:Q'
).properties(width=100)
    c3 = c1|c2
    return c3

def plot_cat_con(nov_funnel):
    categories = list(nov_funnel.category_code.unique())
    top_10_cat_funnel = nov_funnel.loc[nov_funnel['category_code'].isin(categories)]

    cat_dropdown = alt.binding_select(options=categories)
    cat_select = alt.selection_single(fields=['category_code'], bind=cat_dropdown, name="Category")

    c4 = alt.Chart(top_10_cat_funnel,title = 'Novemeber:Conversion for top 10 category_codes').mark_bar().encode(
    x=alt.X('event_type:N',sort = ('view','cart','purchase')),
    y = alt.Y('funnel_value:Q'),

).properties(
    width=200,
    height=300
).resolve_scale(y='independent').add_selection(
    cat_select
).transform_filter(
    cat_select
).properties(title="Select a Category to view Customer Behavior")
    return c4



top_10_brand_sales = generate_base_counts(sales_nov,metrics = 'product_id',grouped = 'brand',top = 10)
top_10_brand_rev = generate_base_sum(sales_nov,metrics = 'price',grouped = 'brand',top = 10)
nov_brand_conversions = conversions(nov,grouped = 'brand',metric = 'user_session',comp = top_10_brand_sales )
```

```
129 nov_brand_funnel = funnel(nov,grouped = 'brand',metric = 'user_session')
130
131
132
133 def plot_top_brand(top_10_brand_sales,top_10_brand_rev):
134     c5 = alt.Chart(top_10_brand_sales,title = 'Top 10 brands by # of Sales').mark_bar().encode(
135     x=alt.X('product_id:Q',title = '# of sales'),
136     y = alt.Y('brand:N',sort = '-x') ,
137 )
138
139     c6 = alt.Chart(top_10_brand_rev,title = 'Total Revenue').mark_text().encode(
140     y=alt.Y('brand:N',axis = None,sort = '-text'),
141     text='price:Q'
142 ).properties(width=100)
143
144     c7 = c5 |c6
145     return c7
146
147
148 def plot_brand_con(nov_brand_funnel):
149     brands = list(nov_brand_funnel.brand.unique())
150     brand_dropdown = alt.binding_select(options=brands)
151     brand_select = alt.selection_single(fields=['brand'], bind=brand_dropdown, name="Brand")
152
153     c8 = alt.Chart(nov_brand_funnel,title = 'Novemeber:Conversion for top 10 brands').mark_bar().encode(
154     x=alt.X('event_type:N',sort = ('view','cart','purchase')),
155     y = alt.Y('funnel_value:Q'),
156
157 ).properties(
158     width=200,
159     height=300
160 ).resolve_scale(y='independent').add_selection(
161     brand_select
162 ).transform_filter(
163     brand_select
164 ).properties(title="Select a Brand to view Customer Behavior")
165     return c8
166
167
168
169 def plot_daily_sale(sales_nov):
170     sales_by_date = sales_nov.groupby(['event_time','brand','category_code']).sum()
171     sales_by_date.reset_index(inplace=True)
172     sales_by_date = sales_by_date[['event_time','brand','price','category_code']]
173     sales_by_date.columns=['event_time','brand','sales','category_code']
174     sales_by_date['event_time'] = sales_by_date['event_time'].apply(lambda x:x.toordinal())
175     sales_by_date['event_time']  = sales_by_date['event_time'] - sales_by_date['event_time'].iloc[0]
176     brands_sbd = list(sales_by_date.brand.unique())
177     brand_dropdown_sbd = alt.binding_select(options=brands_sbd)
178     brand_select_sbd = alt.selection_single(fields=['brand'], bind=brand_dropdown_sbd, name="Brand")
179     categories_sbd = list(sales_by_date.category_code.unique())
180     cat_dropdown_sbd = alt.binding_select(options=categories_sbd)
181     cat_select_sbd = alt.selection_single(fields=['category_code'], bind=cat_dropdown_sbd, name="Category")
182     #Sales by date & brand &cat
183     c9 = alt.Chart(sales_by_date,title = 'Sales by Date').mark_bar().encode(
184     x = alt.X('event_time',scale=alt.Scale(domain=(1, 30))),
185     y = 'sales:Q',
186
187 ).properties(
188     width=300,
189     height=200
190 ).resolve_scale(y='independent').add_selection(
191     cat_select_sbd
192 ).transform_filter(
193     cat_select_sbd
194 ).add_selection(
195     brand_select_sbd
```

```
196 ).transform_filter(
197     brand_select_sbd
198 ).properties(title="Select category and Brand to View Sales by Date")
199     return c9
200
201
202
203 date = pd.DataFrame({'year': [2015, 2015,2015,2015,2015,2015, 2015,2015,2015,2015],
204
205                     'month': [2, 2,2,2,2,2,2,2,2,2],
206
207                     'day': [1,1,1,2,2,2,3,3,3,1]})
208 temp = pd.to_datetime(date)
209
210
211 def plot_general_con(nov):
212     brands_sbd = list(nov.brand.unique())
213     brand_dropdown_sbd = alt.binding_select(options=brands_sbd)
214     brand_select_sbd = alt.selection_single(fields=['brand'], bind=brand_dropdown_sbd, name="Brand")
215
216     categories_sbd = list(nov.category_code.unique())
217     cat_dropdown_sbd = alt.binding_select(options=categories_sbd)
218     cat_select_sbd = alt.selection_single(fields=['category_code'], bind=cat_dropdown_sbd, name="Category")
219
220     master_grouped = nov.groupby(['category_code','brand', 'event_type']).agg({'user_session':'count', 'price':'sum'}
221     master_grouped= master_grouped.reset_index()
222     c10 = alt.Chart(master_grouped, title = 'Cat-> brand test').mark_bar().encode(
223     x=alt.X('event_type:N'),
224     y = alt.Y('user_session:Q',axis=alt.Axis(tickMinStep=1)),
225 ).properties(
226     width=200,
227     height=300
228 ).resolve_scale(y='independent').add_selection(
229     cat_select_sbd
230 ).transform_filter(
231     cat_select_sbd
232 ).add_selection(
233     brand_select_sbd
234 ).transform_filter(
235     brand_select_sbd
236 ).properties(title="Select category and Brand to View Customer Behavior")
237     return c10
```