Classy COB Selection A/B Testing Anlaysis

01 Importing Libraries

In [1]:
```
import numpy as np

import pandas as pd, psycopg2, os
import numpy as np

import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns

import statsmodels.api as sm
import statsmodels.formula.api as smf
from statsmodels.stats.power import NormalIndPower

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

02 Reading SQL Data

In [2]:
```
def query_redshift(sql, user_name, pwd,host):

    host_name= host
    conn = psycopg2.connect(host= host_name, port=5439, dbname='prod', user=use

    cursor = conn.cursor()
    cursor.execute(sql)
    res = cursor.fetchall()

    df = pd.DataFrame(res, columns=[x.name for x in cursor.description])

    cursor.close()
    conn.commit()
    conn.close()

    return df
```

In [3]:
```
username = 'xxxxxx'
password = 'xxxxxxxxx'
```

In [4]:
```
sql_post = '''
with user_base as
(select  t.ab_test_name, t.ab_test_id,
v.ab_test_variant_name,v.ab_test_variant_id,
sa.related_business_id, sa.business_id
from "experiments_svc_prod".ab_tests t
inner join "experiments_svc_prod".ab_test_variants v on t.ab_test_id=v.ab_test_
inner join dwh.all_activities_table aa on aa.ab_test_variant_id=v.ab_test_varia
left join dwh.sources_attributed_table sa on aa.tracking_id=sa.tracking_id
left join next_insurance_prod.cookied_users cu on cu.id=sa.cookied_user_id
where t.ab_test_id = 1401 and eventtime >= '2023-10-12'
group by 1,2,3,4,5,6)

select
   a.tracking_id,
   u.ab_test_name,
   u.ab_test_variant_name,
   a.policy_id,
   a.funnelphase,
    lob,
    s.business_id,
    s.related_business_id,
    eventtime,
    placement, s.cob_name,
    case
        when s.marketing_cob_group is null then nullif(json_extract_path_text(i
            else s.marketing_cob_group end as marketing_cob_group,
    final_quote_status,
    interaction_data,
    json_extract_path_text(source_json_last_related_business_id_before_first_le
    json_extract_path_text(source_json_last_related_business_id_before_first_le
    json_extract_path_text(session_json_last_related_business_id_before_first_
    yearly_premium
from dwh.all_activities_table a
            inner join dwh.sources_attributed_table s
                     on a.tracking_id = s.tracking_id
left join user_base u on u.related_business_id = s.related_business_id
where 1=1
     and eventtime >= '2023-10-12' and eventtime < '2023-11-09' and (a.placemen
and a.tracking_id not in (select distinct aat.tracking_id
from dwh.all_activities_table aat where aat.eventtime between '2023-09-18' and
'''
data = query_redshift(sql_post, username, password, 'redshift-oregon.nextinsura
data = data.fillna('')
```

03 Data Manipulation for Metrics

In [5]:
```
post_data = data
```

In [6]:
```
# add a column for cob to parse out Retail and Food & Bev for post A/B test dat
cob_post = post_data[['related_business_id','marketing_cob_group']].drop_duplic
cob_post = cob_post[cob_post['marketing_cob_group'] != ''].reset_index(drop=Tru
```

```
cob_dic_post = cob_post.set_index('related_business_id')['marketing_cob_group']
post_data['marketing_cob_final'] = post_data['related_business_id'].map(cob_di
```

In [7]:
```
# add a column for A/B test to parse out test and variant related business ids
AB_test = post_data[['related_business_id','ab_test_variant_name']].drop_dupli
AB_test = AB_test[AB_test['ab_test_variant_name'] != ''].reset_index(drop=True
AB_dic = AB_test.set_index('related_business_id')['ab_test_variant_name'].to_d
post_data['ab_test_variant_name_final'] = post_data['related_business_id'].map
```

In [8]:
```
# filter on only Food & Beverage and Retail Data for pre and post data
food = ['FOOD_AND_BEVERAGE', 'Food & beverage', 'Food & beverage - deprecated'
retail = ['RETAIL', 'Retail', 'retail',  'Retail - deprecated']
conditions = [
    post_data['marketing_cob_final'].isin(food),
    post_data['marketing_cob_final'].isin(retail)
]
values = ['Food & Beverage', 'Retail']
post_data['marketing_cob_final2'] = np.select(conditions, values, default='')
```

In [9]:
```
# giving all variations of food & bev and Retial and same name
values_to_match = ['Food & Beverage', 'Retail']
post_data_final = post_data[post_data['marketing_cob_final2'].isin(values_to_ma
post_data_final = post_data_final[post_data_final['ab_test_variant_name_final']
```

In [10]:
```
# creating a Underwriting-success funnelphase; easier to calculate metrics
post_data_final = post_data_final.reset_index()
post_data_final['funnelphasefinal'] = ''
for x in range(len(post_data_final)):
    if post_data_final['funnelphase'][x] == 'Underwriting Processed' and post_
        post_data_final['funnelphasefinal'][x] =  'Underwriting Processed - su
    else:
        post_data_final['funnelphasefinal'][x]= post_data_final['funnelphase']
```

In [11]:
```
#Selecting only columns needed for the regression and droping duplicates to av
selected_columns = ['related_business_id', 'ab_test_variant_name_final','marke
dataset_part1 = post_data_final.loc[:, selected_columns].drop_duplicates()
selected_columns2 = ['related_business_id', 'yearly_premium','policy_id']
dataset_part2 = post_data_final.loc[:, selected_columns2].drop_duplicates()
```

In [12]:
```
#creating pivot table for all the funnel phases
dataset_part1 = dataset_part1.reset_index()
dataset1 = pd.pivot_table(dataset_part1, values='related_business_id',
                          index=['related_business_id', 'ab_test_variant_name_
                          columns='funnelphasefinal',
                          aggfunc='count',
                          fill_value=0)
```

In [13]:
```
#creating pivot table for yearly premium
dataset_part2 = dataset_part2.reset_index()
dataset_part2['yearly_premium'] = pd.to_numeric(dataset_part2['yearly_premium'
dataset2 = pd.pivot_table(dataset_part2,
                           values='yearly_premium',
                           index='related_business_id',
                           aggfunc='sum',
                           fill_value=0)
```

In [14]:
```
# reseting the index and merging yearly premium with other metrics
dataset1 = dataset1.reset_index()
dataset2 = dataset2.reset_index()
dataset = pd.merge(dataset1, dataset2, on = ['related_business_id','related_bus
```

04 Metric Calculations

In [15]:
```
# set up the metrics
dataset = dataset.reset_index()
dataset['Lead_to_Purchase'] = np.where((dataset['Lead'] == 1) & (dataset['Purcl
dataset['Lead_Completion'] = np.where((dataset['Lead'] == 1) & (dataset['click_
dataset['COB_preselected'] = np.where((dataset['cob_classification_COMPLETE - (
dataset['Lead_to_Quote'] = np.where((dataset['Lead'] == 1) & (dataset['Underwri
dataset['Quote_to_Purchase'] = np.where((dataset['Underwriting Processed - suc(
dataset['Start_to_Purchase'] = np.where((dataset['click_next - CLICK lead_form_
dataset['Lead_to_Purchase'] = np.where((dataset['Lead'] == 1) & (dataset['Purcl
dataset['Lead_to_Purchase'] = np.where((dataset['Start_to_Purchase']==1),1,data
dataset['Average_Policy_Price'] = np.where((dataset['Purchase'] == 1) & (datase
```

In [16]:
```
# keep only the columns need for ols regression
columns_to_keep = ['related_business_id', 'ab_test_variant_name_final','market:
       'Start_to_Purchase']
dataset_final = dataset.loc[:, columns_to_keep]
```

In [17]:
```
# We didn't run the A/B test to get all the variables, so using the sample we l
post_data_final.groupby('ab_test_variant_name_final')['related_business_id'].nu
```

Out[17]:
```
ab_test_variant_name_final
2023Sep_auto_cob_selection_control    4344
2023Sep_auto_cob_selection_test       4343
Name: related_business_id, dtype: int64
```

In [18]:
```
dataset_final['ab_test_variant_name_final'] = dataset_final['ab_test_variant_na
```

In [19]:
```
dataset_final['group'] = ''
for x in range(len(dataset_final)):
    if dataset_final['ab_test_variant_name_final'][x] == '2023Sep_auto_cob_sele
        dataset_final['group'][x] = 0
    else:
        dataset_final['group'][x] = 1
```

In [20]:
```
dataset.to_csv('dataset.csv')
```

04 Regressions

Metric 1: % Lead Completion

In [21]:
```
# ols Regression for Lead Completion
formula = "Lead_Completion ~ C(ab_test_variant_name_final)"
model = smf.ols(formula=formula, data=dataset)
outcome = model.fit()
outcome.summary(alpha = 0.1)
```

Out[21]:

OLS Regression Results

Dep. Variable:	Lead_Completion	**R-squared:**	0.011
Model:	OLS	**Adj. R-squared:**	0.011
Method:	Least Squares	**F-statistic:**	96.53
Date:	Wed, 29 Nov 2023	**Prob (F-statistic):**	1.16e-22
Time:	22:37:53	**Log-Likelihood:**	-5581.4
No. Observations:	8687	**AIC:**	1.117e+04
Df Residuals:	8685	**BIC:**	1.118e+04
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.05	0.95]
Intercept	0.7383	0.007	105.757	0.000	0.727	0.750
C(ab_test_variant_name_final)[T.2023Sep_auto_cob_selection_test]	-0.0970	0.010	-9.825	0.000	-0.113	-0.081

Omnibus:	29851.007	**Durbin-Watson:**	1.992
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	1551.574
Skew:	-0.807	**Prob(JB):**	0.00
Kurtosis:	1.702	**Cond. No.**	2.62

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Metric 2: % Lead to Purchase

In [22]:

```
# ols Regression for Lead to Purchase
formula = "Lead_to_Purchase ~ C(ab_test_variant_name_final)"
model = smf.ols(formula=formula, data=dataset)
outcome = model.fit()
outcome.summary(alpha = 0.1)
```

Out[22]: OLS Regression Results

Dep. Variable:	Lead_to_Purchase	**R-squared:**	0.000
Model:	OLS	**Adj. R-squared:**	0.000
Method:	Least Squares	**F-statistic:**	2.998
Date:	Wed, 29 Nov 2023	**Prob (F-statistic):**	0.0834
Time:	22:38:00	**Log-Likelihood:**	-2953.9
No. Observations:	8687	**AIC:**	5912.
Df Residuals:	8685	**BIC:**	5926.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.05	0.95]
Intercept	0.1397	0.005	27.087	0.000	0.131	0.148
C(ab_test_variant_name_final)[T.2023Sep_auto_cob_selection_test]	-0.0126	0.007	-1.731	0.083	-0.025	-0.001

Omnibus:	3335.593	**Durbin-Watson:**	2.021
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	9261.150
Skew:	2.155	**Prob(JB):**	0.00
Kurtosis:	5.647	**Cond. No.**	2.62

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Metric 3: Lead to Quote

In [23]:
```
# ols Regression for Lead to Quote
formula = "Lead_to_Quote ~ C(ab_test_variant_name_final)"
model = smf.ols(formula=formula, data=dataset)
outcome = model.fit()
outcome.summary(alpha = 0.1)
```

Out[23]:

OLS Regression Results

Dep. Variable:	Lead_to_Quote	**R-squared:**	0.003
Model:	OLS	**Adj. R-squared:**	0.002
Method:	Least Squares	**F-statistic:**	22.36
Date:	Wed, 29 Nov 2023	**Prob (F-statistic):**	2.29e-06
Time:	22:38:04	**Log-Likelihood:**	-6192.5
No. Observations:	8687	**AIC:**	1.239e+04
Df Residuals:	8685	**BIC:**	1.240e+04
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.05	0.95]
Intercept	0.4491	0.007	59.967	0.000	0.437	0.461
C(ab_test_variant_name_final)[T.2023Sep_auto_cob_selection_test]	-0.0501	0.011	-4.729	0.000	-0.068	-0.033

Omnibus:	32050.830	**Durbin-Watson:**	1.996
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	1436.501
Skew:	0.306	**Prob(JB):**	0.00
Kurtosis:	1.104	**Cond. No.**	2.62

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Metric 4: Quote to Purchase

In [24]:

```
# ols Regression for Quote to Purchase
formula = "Quote_to_Purchase ~ C(ab_test_variant_name_final)"
model = smf.ols(formula=formula, data=dataset)
outcome = model.fit()
outcome.summary(alpha = 0.1)
```

Out[24]:

OLS Regression Results

Dep. Variable:	Quote_to_Purchase	**R-squared:**	0.000
Model:	OLS	**Adj. R-squared:**	0.000
Method:	Least Squares	**F-statistic:**	2.665
Date:	Wed, 29 Nov 2023	**Prob (F-statistic):**	0.103
Time:	22:38:08	**Log-Likelihood:**	-2976.1
No. Observations:	8687	**AIC:**	5956.
Df Residuals:	8685	**BIC:**	5970.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.05	0.95]
Intercept	0.1402	0.005	27.106	0.000	0.132	0.149
C(ab_test_variant_name_final) [T.2023Sep_auto_cob_selection_test]	-0.0119	0.007	-1.632	0.103	-0.024	9.19e-05

Omnibus:	3312.103	**Durbin-Watson:**	2.023
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	9115.342
Skew:	2.145	**Prob(JB):**	0.00
Kurtosis:	5.604	**Cond. No.**	2.62

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Metric 5: Start to Purchase

In [25]:
```
# ols Regression for Start to Purchase
formula = "Start_to_Purchase ~ C(ab_test_variant_name_final)"
model = smf.ols(formula=formula, data=dataset)
outcome = model.fit()
outcome.summary(alpha = 0.1)
```

Out[25]:

OLS Regression Results

Dep. Variable:	Start_to_Purchase	**R-squared:**	0.000
Model:	OLS	**Adj. R-squared:**	0.000
Method:	Least Squares	**F-statistic:**	2.787
Date:	Wed, 29 Nov 2023	**Prob (F-statistic):**	0.0951
Time:	22:38:12	**Log-Likelihood:**	-2947.6
No. Observations:	8687	**AIC:**	5899.
Df Residuals:	8685	**BIC:**	5913.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.05**	**0.95]**
Intercept	0.1393	0.005	27.017	0.000	0.131	0.148
C(ab_test_variant_name_final)[T.2023Sep_auto_cob_selection_test]	-0.0122	0.007	-1.669	0.095	-0.024	-0.000

Omnibus:	3342.532	**Durbin-Watson:**	2.022
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	9304.470
Skew:	2.158	**Prob(JB):**	0.00
Kurtosis:	5.660	**Cond. No.**	2.62

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Metric 6: Average Policy Price

In [26]:

```
# ols Regression for Average Policy Price
formula = "Average_Policy_Price ~ C(ab_test_variant_name_final)"
model = smf.ols(formula=formula, data=dataset)
outcome = model.fit()
outcome.summary()
```

Out[26]:

OLS Regression Results

Dep. Variable:	Average_Policy_Price	**R-squared:**	0.000
Model:	OLS	**Adj. R-squared:**	0.000
Method:	Least Squares	**F-statistic:**	2.035
Date:	Wed, 29 Nov 2023	**Prob (F-statistic):**	0.154
Time:	22:38:18	**Log-Likelihood:**	-65839.
No. Observations:	8687	**AIC:**	1.317e+05
Df Residuals:	8685	**BIC:**	1.317e+05
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	103.4681	7.184	14.402	0.000	89.385	117.551
C(ab_test_variant_name_final) [T.2023Sep_auto_cob_selection_test]	-14.4946	10.161	-1.427	0.154	-34.412	5.423

Omnibus:	18378.878	**Durbin-Watson:**	1.972
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	146355687.485
Skew:	18.108	**Prob(JB):**	0.00
Kurtosis:	637.848	**Cond. No.**	2.62

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Metric 7: COB Pre-Selected

In [27]:
```
dataset['COB_preselected'].value_counts()
```

Out[27]:
```
0    7665
1    1022
Name: COB_preselected, dtype: int64
```

05 Package Distributions

In [28]:
```
# For the policies quoted, we are looking at average distribution of package
sql2 = '''
select*
from
(select  t.ab_test_name, t.ab_test_id,
case
 when v.ab_test_variant_name = '2023Sep_auto_cob_selection_control' then 'cont
 when v.ab_test_variant_name = '2023Sep_auto_cob_selection_test' then 'variant
 else '' end as ab_test_variant_name ,
v.ab_test_variant_id,
```

```
sa.related_business_id, sa.business_id, qpm.cob_group, qpm.num_of_employees, qp
qpm.year_business_started, qpm.highest_status_package,
rank() over(partition by sa.related_business_id order by aa.eventtime, qpm.crea
from "experiments_svc_prod".ab_tests t
inner join "experiments_svc_prod".ab_test_variants v on t.ab_test_id=v.ab_test_
inner join dwh.all_activities_table aa on aa.ab_test_variant_id=v.ab_test_varia
left join dwh.sources_attributed_table sa on aa.tracking_id=sa.tracking_id
left join next_insurance_prod.cookied_users cu on cu.id=sa.cookied_user_id
left join dwh.quotes_policies_mlob qpm on qpm.related_business_id = sa.related_
where t.ab_test_id = 1401 and eventtime >= '2023-10-12'
group by t.ab_test_name, t.ab_test_id,
v.ab_test_variant_name,v.ab_test_variant_id,
sa.related_business_id, sa.business_id, qpm.cob_group,
qpm.num_of_employees, qpm.num_of_owners,
qpm.payroll_in_next_12_months, qpm.years_of_experience,
qpm.year_business_started,
qpm.highest_status_package,
aa.eventtime,
qpm.creation_time
) z where Rank = 1;
'''
package_data = query_redshift(sql2, username, password, 'redshift-oregon.nexti
package_data = package_data.fillna('')
```

In [29]:
```
package_data_quoted = package_data[package_data['highest_status_package']!= '']
```

In [30]:
```
pivot_table = pd.pivot_table(package_data_quoted, values='related_business_id'
pivot_table
```

Out[30]:

highest_status_package	basic	basicTria	pro	proPlus	proPlusTria	proTria
ab_test_variant_name						
control	1772	2	380	343	14	14
variant	1668	2	371	340	20	4

In [31]:
```
pivot_table['BASIC'] = pivot_table['basic'] + pivot_table['basicTria']
pivot_table['PRO'] = pivot_table['pro']  + pivot_table['proTria']
pivot_table['PROPLUS'] = pivot_table['proPlus'] +  pivot_table['proPlusTria']
```

In [32]:
```
pivot_table
```

Out[32]:

highest_status_package	basic	basicTria	pro	proPlus	proPlusTria	proTria	BASIC	PRO	PRO
ab_test_variant_name									
control	1772	2	380	343	14	14	1774	394	
variant	1668	2	371	340	20	4	1670	375	

In [33]:
```
pivot_table_combined = pivot_table[['BASIC','PRO','PROPLUS']]
```

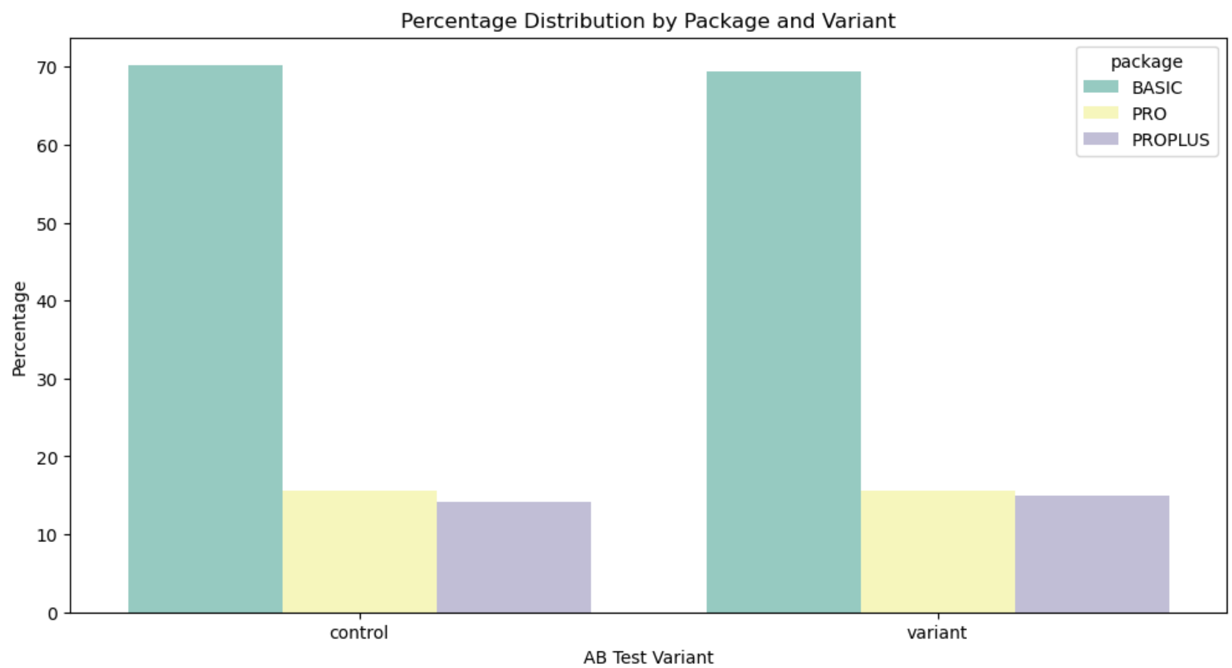
In [34]:
```
percentage_pivot_table = pivot_table_combined.apply(lambda x: (x / x.sum()) * 1
percentage_pivot_table = percentage_pivot_table.reset_index()
percentage_pivot_table
```

Out[34]:

highest_status_package	ab_test_variant_name	BASIC	PRO	PROPLUS
0	control	70.257426	15.603960	14.138614
1	variant	69.438669	15.592516	14.968815

In [35]:
```
melted_table = pd.melt(percentage_pivot_table, id_vars=['ab_test_variant_name']
```

In [36]:
```
plt.figure(figsize=(12, 6))
sns.barplot(x='ab_test_variant_name', y='percentage', hue='package', data=melte
plt.title('Percentage Distribution by Package and Variant')
plt.xlabel('AB Test Variant')
plt.ylabel('Percentage')
plt.show()
```



06 Difference in Difference for Mobile Impact

In [37]:
```
dateset_dff = dataset
```

In [38]:
```
selected_columns3 = ['related_business_id', 'device']
dataset_device = post_data_final.loc[:, selected_columns3].drop_duplicates().re
device_dic = dataset_device.set_index('related_business_id')['device'].to_dict
dateset_dff['device'] = dateset_dff['related_business_id'].map(device_dic)
```

In [39]:
```
dateset_dff['Device_Updated'] = np.where(dateset_dff['device'] == 'desktop', 0
dateset_dff['Treatment'] = np.where(dateset_dff['ab_test_variant_name_final'] =
dateset_dff['TreatmentDevice'] = dateset_dff['Treatment'] * dateset_dff['Device
```

In [40]:
```
# Difference in Difference for Lead Completion
formula = "Lead_Completion ~ Treatment * Device_Updated"
model = smf.ols(formula=formula, data=dateset_dff)
outcome = model.fit()
outcome.summary()
```

Out[40]:

OLS Regression Results

Dep. Variable:	Lead_Completion	**R-squared:**	0.021
Model:	OLS	**Adj. R-squared:**	0.020
Method:	Least Squares	**F-statistic:**	61.51
Date:	Wed, 29 Nov 2023	**Prob (F-statistic):**	2.40e-39
Time:	22:39:18	**Log-Likelihood:**	-5538.0
No. Observations:	8687	**AIC:**	1.108e+04
Df Residuals:	8683	**BIC:**	1.111e+04
Df Model:	3		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	0.7971	0.011	70.630	0.000	0.775	0.819
Treatment	-0.0983	0.016	-6.195	0.000	-0.129	-0.067
Device_Updated	-0.0947	0.014	-6.614	0.000	-0.123	-0.067
Treatment:Device_Updated	0.0008	0.020	0.042	0.967	-0.039	0.040

Omnibus:	19319.892	**Durbin-Watson:**	1.988
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	1498.890
Skew:	-0.793	**Prob(JB):**	0.00
Kurtosis:	1.725	**Cond. No.**	7.84

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [41]:

```
# Difference in Difference for Lead to Purchase
formula = "Lead_to_Purchase ~ Treatment * Device_Updated"
model = smf.ols(formula=formula, data=dateset_dff)
outcome = model.fit()
outcome.summary()
```

Out[41]:

OLS Regression Results

Dep. Variable:	Lead_to_Purchase	**R-squared:**	0.001
Model:	OLS	**Adj. R-squared:**	0.001
Method:	Least Squares	**F-statistic:**	3.281
Date:	Wed, 29 Nov 2023	**Prob (F-statistic):**	0.0200
Time:	22:39:27	**Log-Likelihood:**	-2950.4
No. Observations:	8687	**AIC:**	5909.
Df Residuals:	8683	**BIC:**	5937.
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	0.1531	0.008	18.273	0.000	0.137	0.170
Treatment	-0.0152	0.012	-1.294	0.196	-0.038	0.008
Device_Updated	-0.0215	0.011	-2.024	0.043	-0.042	-0.001
Treatment:Device_Updated	0.0040	0.015	0.265	0.791	-0.025	0.033

Omnibus:	3331.185	**Durbin-Watson:**	2.021
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	9237.913
Skew:	2.153	**Prob(JB):**	0.00
Kurtosis:	5.643	**Cond. No.**	7.84

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [42]:

```
# Difference in Difference for Lead to Quote
formula = "Lead_to_Quote ~ Treatment * Device_Updated"
model = smf.ols(formula=formula, data=dataset_dff)
outcome = model.fit()
outcome.summary()
```

Out[42]:

OLS Regression Results

Dep. Variable:	Lead_to_Quote	**R-squared:**	0.013
Model:	OLS	**Adj. R-squared:**	0.012
Method:	Least Squares	**F-statistic:**	37.17
Date:	Wed, 29 Nov 2023	**Prob (F-statistic):**	7.34e-24
Time:	22:39:36	**Log-Likelihood:**	-6148.2
No. Observations:	8687	**AIC:**	1.230e+04
Df Residuals:	8683	**BIC:**	1.233e+04
Df Model:	3		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	0.5176	0.012	42.755	0.000	0.494	0.541
Treatment	-0.0613	0.017	-3.600	0.000	-0.095	-0.028
Device_Updated	-0.1103	0.015	-7.179	0.000	-0.140	-0.080
Treatment:Device_Updated	0.0167	0.022	0.772	0.440	-0.026	0.059

Omnibus:	32970.626	**Durbin-Watson:**	1.993
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	1379.296
Skew:	0.303	**Prob(JB):**	3.09e-300
Kurtosis:	1.144	**Cond. No.**	7.84

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [43]:
```
# Difference in Difference for Quote to Purchase
formula = "Quote_to_Purchase ~ Treatment * Device_Updated"
model = smf.ols(formula=formula, data=dateset_dff)
outcome = model.fit()
outcome.summary()
```

Out[43]:

OLS Regression Results

Dep. Variable:	Quote_to_Purchase	**R-squared:**	0.001
Model:	OLS	**Adj. R-squared:**	0.001
Method:	Least Squares	**F-statistic:**	3.299
Date:	Wed, 29 Nov 2023	**Prob (F-statistic):**	0.0195
Time:	22:39:39	**Log-Likelihood:**	-2972.5
No. Observations:	8687	**AIC:**	5953.
Df Residuals:	8683	**BIC:**	5981.
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	0.1537	0.008	18.299	0.000	0.137	0.170
Treatment	-0.0141	0.012	-1.191	0.234	-0.037	0.009
Device_Updated	-0.0218	0.011	-2.041	0.041	-0.043	-0.001
Treatment:Device_Updated	0.0032	0.015	0.211	0.833	-0.026	0.033

Omnibus:	3307.457	**Durbin-Watson:**	2.023
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	9091.141
Skew:	2.142	**Prob(JB):**	0.00
Kurtosis:	5.600	**Cond. No.**	7.84

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [44]:

```
# Difference in Difference for Start to Purchase
formula = "Start_to_Purchase ~ Treatment * Device_Updated"
model = smf.ols(formula=formula, data=dateset_dff)
outcome = model.fit()
outcome.summary()
```

Out[44]:

OLS Regression Results

Dep. Variable:	Start_to_Purchase	**R-squared:**	0.001
Model:	OLS	**Adj. R-squared:**	0.001
Method:	Least Squares	**F-statistic:**	3.184
Date:	Wed, 29 Nov 2023	**Prob (F-statistic):**	0.0228
Time:	22:39:46	**Log-Likelihood:**	-2944.2
No. Observations:	8687	**AIC:**	5896.
Df Residuals:	8683	**BIC:**	5925.
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	0.1525	0.008	18.214	0.000	0.136	0.169
Treatment	-0.0146	0.012	-1.243	0.214	-0.038	0.008
Device_Updated	-0.0213	0.011	-2.003	0.045	-0.042	-0.000
Treatment:Device_Updated	0.0037	0.015	0.249	0.803	-0.026	0.033

Omnibus:	3338.166	**Durbin-Watson:**	2.022
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	9281.360
Skew:	2.156	**Prob(JB):**	0.00
Kurtosis:	5.656	**Cond. No.**	7.84

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

07 Experimental Analysis Tables

In [45]:
```
Table_of_outcomes = dataset.groupby('ab_test_variant_name_final').agg({'click_n
                                                'Lead': 'sum',
                                                'Underwriting Processed - succes
                                                'Purchase': 'sum',
                                                'yearly_premium': 'mean',
                                              }).reset_index()
```

In [46]:
```
Table_of_outcomes
```

Out[46]:

	ab_test_variant_name_final	click_next - CLICK lead_form_industry	Lead	Underwriting Processed - success	Purchase	yearly
0	2023Sep_auto_cob_selection_control	4327	3212	1972	609	
1	2023Sep_auto_cob_selection_test	4325	2788	1756	558	

In [47]:
```
Stats = dataset.groupby('ab_test_variant_name_final').agg({'Lead_Completion':
                                                          'Lead_to_Quote': 'mean',
                                                          'Lead_to_Purchase': 'mean',
                                                          'Start_to_Purchase': 'mean'
                                                         }).reset_index()
```

In [48]:
```
Stats
```

Out[48]:

	ab_test_variant_name_final	Lead_Completion	Lead_to_Quote	Lead_to_Purchase	Sta
0	2023Sep_auto_cob_selection_control	0.738260	0.449125	0.139733	
1	2023Sep_auto_cob_selection_test	0.641262	0.399033	0.127101	

In [ ]: