Model Summary

Decision tree resembles a flow chart. The key of this algorithm is impurity reduction, which means the tree tries to increase the node homogeneity in each node. The tree starts with the feature that can reduce the most impurity, then continue until the tree stops growing. If the tree stops splitting, it forms a 'leaf node'. In Regression Tree, the predicted value is the mean value of all observations in the leaf that it belongs to. The impurity measure is node's variance.

Similar to KNN, Tree algorithm also has the risk of overfitting to training data. We can alleviate this problem by pruning the tree by implementing stopping rules. Tree's strength is that it and can help us visualize how decisions are made and the result of each decision. Because of this characteristic, we have better understanding of what features are more important in determining the outcome.

The final regression tree (becnhmark, or original Product Type) was pruned with maximum depth of 12 and minimum observations to be leaf node as 150 observations.

The most important variable is number of OTB bookings as it showed up in 3 top levels of the tree. The fact that OTB showed up on multiple leaf in the tree indicated that OTB had a non-linear relationship with Cancellation Rate. The importance scores (table below) also showed that OTB is the most important feature, which mean this feature decreased impurity (measured by variance) the most. In our tree, OTB decreased 37% of total impurity.
Product Types at the extreme ends of cancellation behavior also showed themselves as important factors in prediction. They are Product Types with extremely high cancellation rate (Tactical Marketing, Wholesale) and with extremely low cancellation rate (Fenced, Opaque, Other).

Cumulative Gross Bookings also showed up as an important feature. Even though this variable might be close to OTB, it differs from OTB in an important way that Cumulative Gross Bookings indicated demand level while OTB took cancellations into account. This understanding of Cumulative Gross Bookings is important for model interpretation.

```
# Basic packages
import pandas as pd
import numpy as np

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

# Display multiple results in 1 block
from IPython.display import display

from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from math import sqrt

from sklearn.tree import DecisionTreeRegressor
from sklearn.tree import export_graphviz

from sklearn.externals.six import StringIO
from IPython.display import Image
from graphviz import Source
import pydotplus
```

```
raw_train_nyc = pd.read_csv('../treated data/train_nyc.csv', index_col = 0)
raw_test_nyc = pd.read_csv('../treated data/test_nyc.csv', index_col = 0)
```

```
train_nyc = raw_train_nyc.copy()
test_nyc = raw_test_nyc.copy()

# Create X, Y for train test set
train_nyc_Y = train_nyc[['cxl_rate']].copy()
train_nyc_Y = train_nyc_Y['cxl_rate'].to_numpy()

train_nyc_X = train_nyc.loc[:, train_nyc.columns != 'cxl_rate'].copy()
test_nyc_X = test_nyc.loc[:, test_nyc.columns != 'cxl_rate'].copy()
```

```
display(raw_train_nyc.columns)
display(train_nyc_X.columns)
```

```
Index(['knn_pred', 'dow_regroup', 'last_week', 'product_type_regroup4',
       'product_type_regroup3', 'product_type_regroup2',
       'product_type_regroup', 'product_type', 'stay_dt', 'dow', 'booking_dt',
       'days_prior', 'daily_gross_bookings', 'daily_gross_rev',
       'daily_cxl_bookings', 'daily_cxl_rev', 'daily_net_bookings',
       'daily_net_rev', 'cummulative_gross_bookings', 'cummulative_gross_rev',
       'cummulative_cxl_bookings', 'cummulative_cxl_rev', 'OTB', 'OTB_rev',
       'OTB_to_be_cxl', 'OTB_rev_to_be_cxl', 'OTB_to_survive',
       'OTB_rev_to_survive', 'room_price', 'days_prior_cat', 'cxl_rate',
       'naive_cxl_rate', 'naive_survive_pred', 'lag1', 'lag2', 'lag3', 'lag4',
       'lag5', 'lag6', 'lag7', 'lag10'],
      dtype='object')



Index(['knn_pred', 'dow_regroup', 'last_week', 'product_type_regroup4',
       'product_type_regroup3', 'product_type_regroup2',
       'product_type_regroup', 'product_type', 'stay_dt', 'dow', 'booking_dt',
       'days_prior', 'daily_gross_bookings', 'daily_gross_rev',
       'daily_cxl_bookings', 'daily_cxl_rev', 'daily_net_bookings',
       'daily_net_rev', 'cummulative_gross_bookings', 'cummulative_gross_rev',
       'cummulative_cxl_bookings', 'cummulative_cxl_rev', 'OTB', 'OTB_rev',
       'OTB_to_be_cxl', 'OTB_rev_to_be_cxl', 'OTB_to_survive',
       'OTB_rev_to_survive', 'room_price', 'days_prior_cat', 'naive_cxl_rate',
       'naive_survive_pred', 'lag1', 'lag2', 'lag3', 'lag4', 'lag5', 'lag6',
       'lag7', 'lag10'],
      dtype='object')
```

```
display(train_nyc_X.shape)
display(test_nyc_X.shape)
display(train_nyc_Y.shape)
```

```
(43920, 40)



(13560, 40)
```

```
(43920, 1)
```

Normalize / Scale data

Select variables for tree (omit monetary values)

```
def preparexy (df, method):
    # deal with categorical variables and drop useless columns
    df = df[[method,'dow', 'days_prior', 'daily_gross_bookings','daily_cxl_bookings',
'daily_net_bookings',\
            'cummulative_gross_bookings','cummulative_cxl_bookings','OTB',
'days_prior_cat']]
    df = pd.get_dummies(df)

    return df
```

```
train_nyc_X_g1 = preparexy(train_nyc, 'product_type_regroup')
test_nyc_X_g1 = preparexy(test_nyc, 'product_type_regroup')

train_nyc_X_g2 = preparexy(train_nyc, 'product_type_regroup2')
test_nyc_X_g2 = preparexy(test_nyc, 'product_type_regroup2')

train_nyc_X_g3 = preparexy(train_nyc, 'product_type_regroup3')
test_nyc_X_g3 = preparexy(test_nyc, 'product_type_regroup3')

train_nyc_X_g4 = preparexy(train_nyc, 'product_type_regroup4')
test_nyc_X_g4 = preparexy(test_nyc, 'product_type_regroup4')

train_nyc_X_bm = preparexy(train_nyc, 'product_type')
test_nyc_X_bm = preparexy(test_nyc, 'product_type')
```

Metrics

```
from sklearn.metrics import mean_absolute_error

#MAE
def mae_table( g ):
    output = round(mean_absolute_error(g['predict_OTB_to_survive'],
g['OTB_to_survive']),4)
    return pd.Series( dict(mae = output ) )
def get_mae(test_data, original_data,model, name):
    test_predictions = model.predict(test_data).flatten()
    test = original_data.copy()
    test['pred'] = test_predictions
    test['predict_OTB_to_survive'] = test['OTB'] -  test['pred'] * test['OTB']

    output = test.groupby( 'days_prior_cat' ).apply( mae_table
```

```
).reset_index().rename(columns={'mae': name})
    return output

# MAPE
def mape(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean(np.abs((y_true - y_pred) / y_true))

def mape_table( g ):
    output = round(mape(g['predict_OTB_to_survive'], g['OTB_to_survive']),4)
    return pd.Series( dict(mape = output ) )

def get_mape(test_data, original_data, model, name):
    test_predictions = model.predict(test_data).flatten()
    test = original_data.copy()
    test['pred'] = test_predictions
    test['predict_OTB_to_survive'] = test['OTB'] -  test['pred'] * test['OTB']

    output = test[test['OTB_to_survive'] != 0].groupby( 'days_prior_cat' )\
                                                .apply( mape_table
).reset_index().rename(columns={'mape': name})
    return output

#MASE
def mase(y_true, y_pred, y_naive):
    y_true, y_pred, y_naive = np.array(y_true), np.array(y_pred), np.array(y_naive)
    return np.sum(np.abs(y_true - y_pred)) / np.sum(np.abs(y_true - y_naive))

def mase_table(g):
    output = round(mase(g['predict_OTB_to_survive'], g['OTB_to_survive'],
g['naive_survive_pred']),4)
    return pd.Series( dict(mase = output ) )

def get_mase(test_data, original_data, model, name):
    test_predictions = model.predict(test_data).flatten()
    test = original_data.copy()
    test['pred'] = test_predictions
    test['predict_OTB_to_survive'] = test['OTB'] -  test['pred'] * test['OTB']

    output = test.groupby( 'days_prior_cat' ).apply( mase_table
).reset_index().rename(columns={'mase': name})
    return output
```

Models

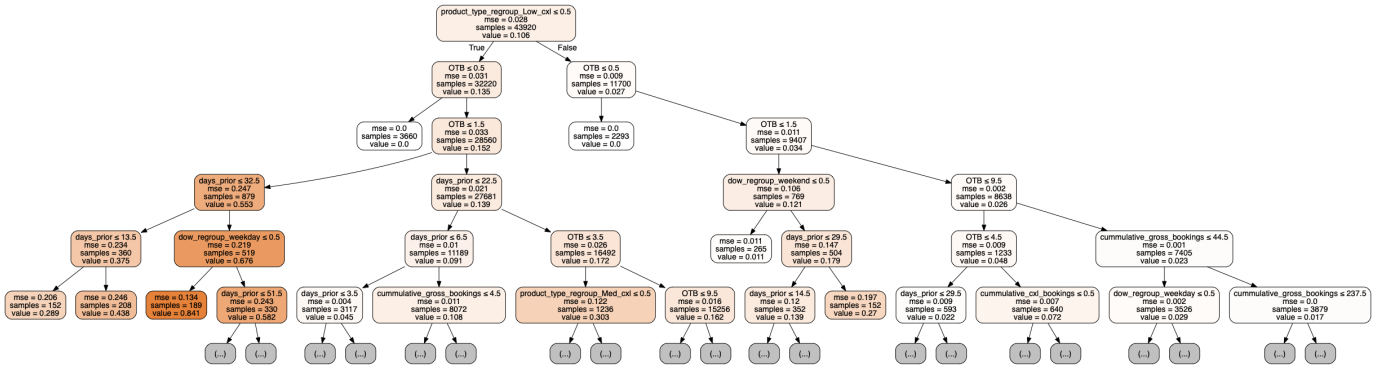
```
def visualize(model,df):
    dot_data = StringIO()
    export_graphviz(model, out_file=dot_data, max_depth=5,feature_names=df.columns,
    filled=True, rounded=True,special_characters=True)
    graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
    return Image(graph.create_png())
```

Group 1

```
model_g1 = DecisionTreeRegressor(min_samples_leaf=150,
                                 max_depth = 11)
model_g1.fit(train_nyc_X_g1, train_nyc_Y)
```

```
DecisionTreeRegressor(criterion='mse', max_depth=11, max_features=None,
                      max_leaf_nodes=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=150,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      presort=False, random_state=None, splitter='best')
```

```
visualize(model_g1, train_nyc_X_g1)
```



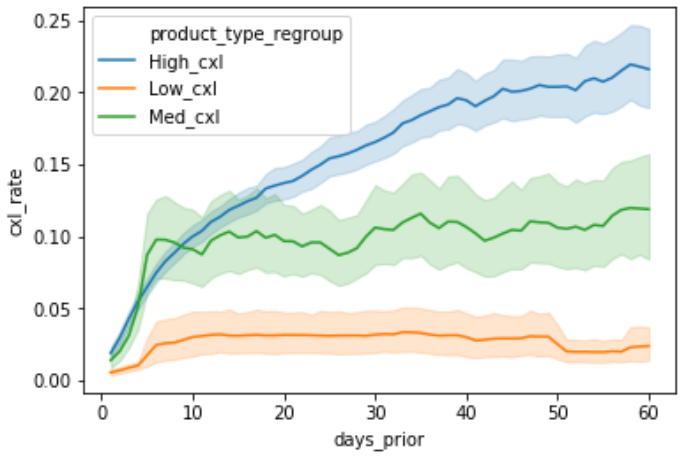

```
mae_g1_in = get_mae(train_nyc_X_g1, raw_train_nyc, model_g1, 'mae_g1_in')
mape_g1_in= get_mape(train_nyc_X_g1, raw_train_nyc, model_g1, 'mape_g1_in')
mase_g1_in = get_mase(train_nyc_X_g1, raw_train_nyc, model_g1, 'mase_g1_in')

mae_g1_out = get_mae(test_nyc_X_g1, raw_test_nyc, model_g1, 'mae_g1_out')
mape_g1_out = get_mape(test_nyc_X_g1, raw_test_nyc, model_g1, 'mape_g1_out')
mase_g1_out = get_mase(test_nyc_X_g1, raw_test_nyc, model_g1, 'mase_g1_out')

mae_g1_in.set_index('days_prior_cat').\
join(mape_g1_in.set_index('days_prior_cat')).\
join(mase_g1_in.set_index('days_prior_cat')).\
join(mae_g1_out.set_index('days_prior_cat')).\
join(mape_g1_out.set_index('days_prior_cat')).\
join(mase_g1_out.set_index('days_prior_cat')).reset_index()
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

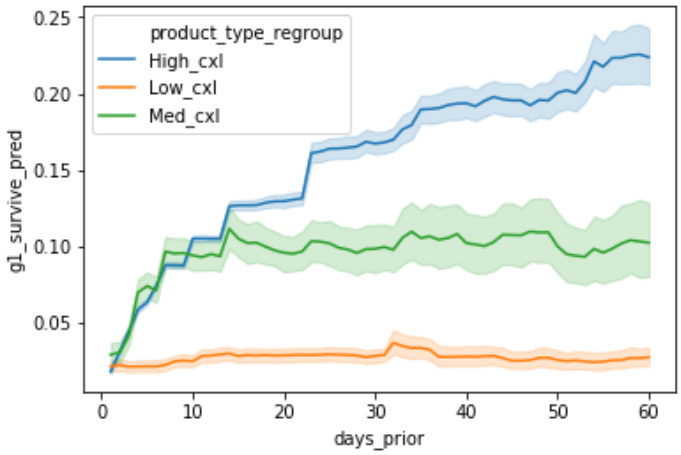
	days_prior_cat	mae_g1_in	mape_g1_in	mase_g1_in	mae_g1_out	mape_g1_out	mase_g1_out

	days_prior_cat	mae_g1_in	mape_g1_in	mase_g1_in	mae_g1_out	mape_g1_out	mase_g1_out
0	Day 01-07	1.5156	0.0414	0.2326	1.8624	0.0305	0.2202
1	Day 08-14	1.8424	0.0606	0.5790	2.4137	0.0458	0.4923
2	Day 15-20	1.8466	0.0692	0.7018	2.3693	0.0538	0.5543
3	Day 21-27	1.7449	0.0808	0.7137	2.7888	0.0766	0.6312
4	Day 28-60	1.1340	0.1165	0.6454	2.2701	0.1252	0.6175

```
# Actual value plot
sns.lineplot(x="days_prior", y="cxl_rate",hue = 'product_type_regroup',
data=train_nyc);
```




```
# Add prediction to train_nyc
train_nyc['g1_survive_pred'] = model_g1.predict(train_nyc_X_g1).flatten()
# Predicted value plot
sns.lineplot(x="days_prior", y="g1_survive_pred",hue = 'product_type_regroup',
data=train_nyc);
```

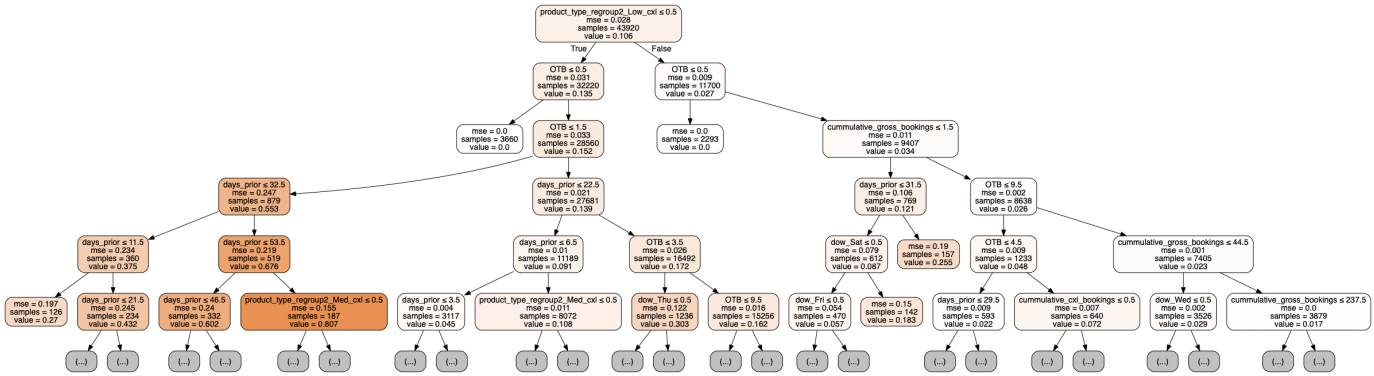


Group 2

```
model_g2 = DecisionTreeRegressor(min_samples_leaf=90,
                                 max_depth = 11)
model_g2.fit(train_nyc_X_g2, train_nyc_Y)
```

```
DecisionTreeRegressor(criterion='mse', max_depth=11, max_features=None,
                      max_leaf_nodes=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=90,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      presort=False, random_state=None, splitter='best')
```

```
visualize(model_g2, train_nyc_X_g2)
```



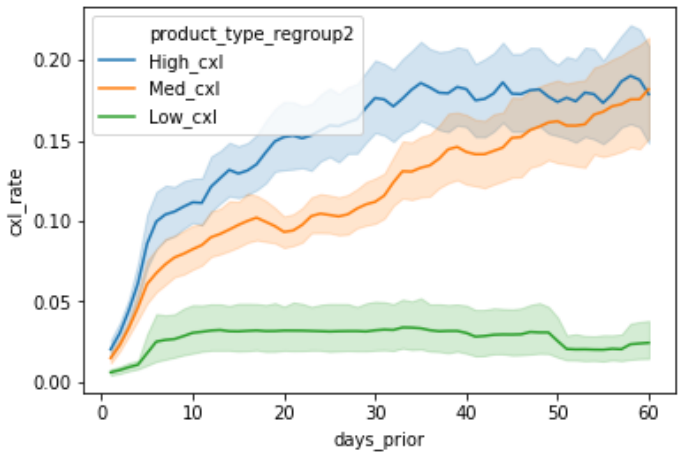

```
mae_g2_in = get_mae(train_nyc_X_g2, raw_train_nyc, model_g2, 'mae_g2_in')
mape_g2_in= get_mape(train_nyc_X_g2, raw_train_nyc, model_g2, 'mape_g2_in')
mase_g2_in = get_mase(train_nyc_X_g2, raw_train_nyc, model_g2, 'mase_g2_in')

mae_g2_out = get_mae(test_nyc_X_g2, raw_test_nyc, model_g2, 'mae_g2_out')
mape_g2_out = get_mape(test_nyc_X_g2, raw_test_nyc, model_g2, 'mape_g2_out')
mase_g2_out = get_mase(test_nyc_X_g2, raw_test_nyc, model_g2, 'mase_g2_out')

mae_g2_in.set_index('days_prior_cat').\
join(mape_g2_in.set_index('days_prior_cat')).\
join(mase_g2_in.set_index('days_prior_cat')).\
join(mae_g2_out.set_index('days_prior_cat')).\
join(mape_g2_out.set_index('days_prior_cat')).\
join(mase_g2_out.set_index('days_prior_cat')).reset_index()
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

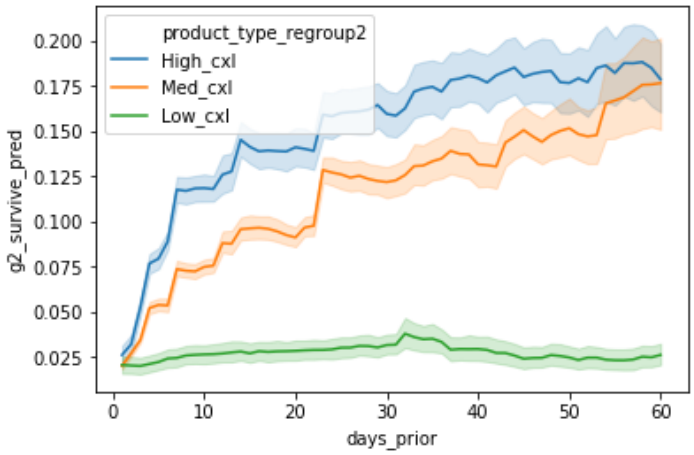
	days_prior_cat	mae_g2_in	mape_g2_in	mase_g2_in	mae_g2_out	mape_g2_out	mase_g2_out

	days_prior_cat	mae_g2_in	mape_g2_in	mase_g2_in	mae_g2_out	mape_g2_out	mase_g2_out
0	Day 01-07	1.5551	0.0396	0.2444	2.0244	0.0311	0.2459
1	Day 08-14	1.8515	0.0590	0.5754	2.6219	0.0478	0.5470
2	Day 15-20	1.7488	0.0638	0.6480	2.4186	0.0564	0.5766
3	Day 20-27	1.7525	0.0812	0.7295	2.6999	0.0772	0.6467
4	Day 28-60	1.1848	0.1153	0.6613	2.2721	0.1560	0.6305

```
# Actual value plot
sns.lineplot(x="days_prior", y="cxl_rate",hue = 'product_type_regroup2',
data=train_nyc);
```




```
# Add prediction to train_nyc
train_nyc['g2_survive_pred'] = model_g2.predict(train_nyc_X_g2).flatten()
# Predicted value plot
sns.lineplot(x="days_prior", y="g2_survive_pred",hue = 'product_type_regroup2',
data=train_nyc);
```

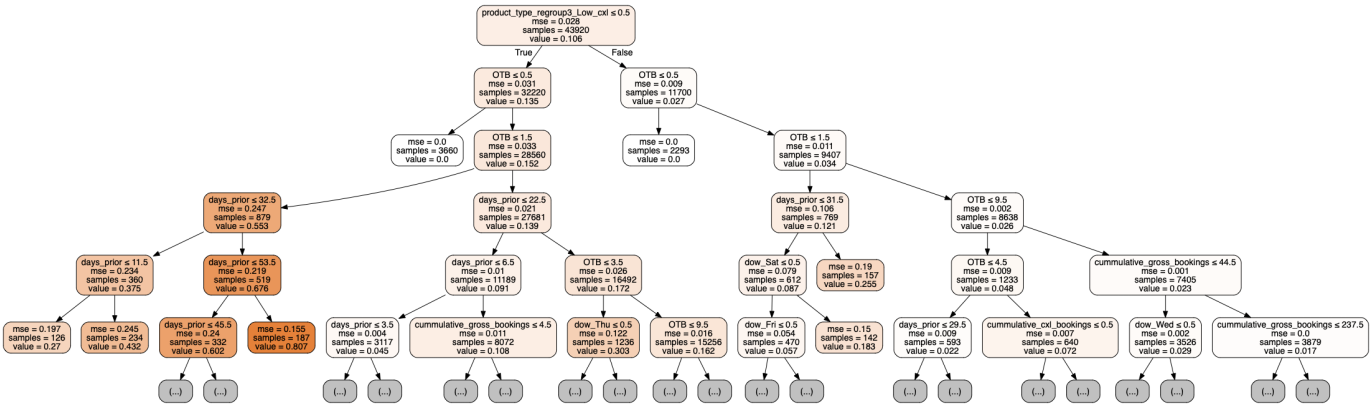


Group 4

```
model_g3 = DecisionTreeRegressor(min_samples_leaf=120,
                                 max_depth = 10)
model_g3.fit(train_nyc_X_g3, train_nyc_Y)
```

```
DecisionTreeRegressor(criterion='mse', max_depth=10, max_features=None,
                      max_leaf_nodes=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=120,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      presort=False, random_state=None, splitter='best')
```

```
visualize(model_g3, train_nyc_X_g3)
```

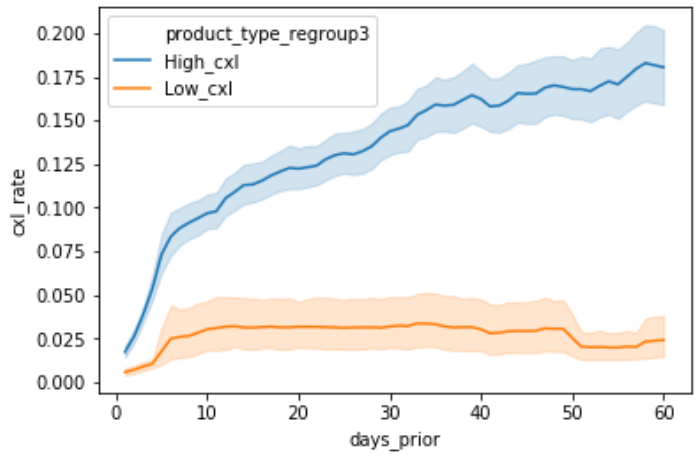



```
mae_g3_in = get_mae(train_nyc_X_g3, raw_train_nyc, model_g3, 'mae_g3_in')
mape_g3_in= get_mape(train_nyc_X_g3, raw_train_nyc, model_g3, 'mape_g3_in')
mase_g3_in = get_mase(train_nyc_X_g3, raw_train_nyc, model_g3, 'mase_g3_in')

mae_g3_out = get_mae(test_nyc_X_g3, raw_test_nyc, model_g3, 'mae_g3_out')
mape_g3_out = get_mape(test_nyc_X_g3, raw_test_nyc, model_g3, 'mape_g3_out')
mase_g3_out = get_mase(test_nyc_X_g3, raw_test_nyc, model_g3, 'mase_g3_out')

mae_g3_in.set_index('days_prior_cat').\
join(mape_g3_in.set_index('days_prior_cat')).\
join(mase_g3_in.set_index('days_prior_cat')).\
join(mae_g3_out.set_index('days_prior_cat')).\
join(mape_g3_out.set_index('days_prior_cat')).\
join(mase_g3_out.set_index('days_prior_cat')).reset_index()
```

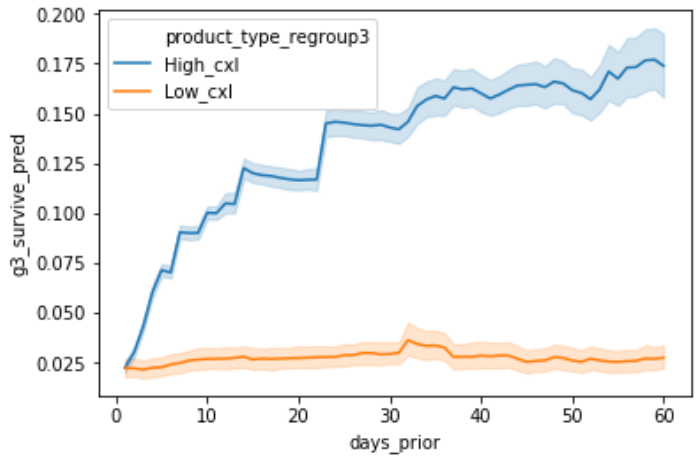
```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	days_prior_cat	mae_g3_ln	mape_g3_ln	mase_g3_ln	mae_g3_out	mape_g3_out	mase_g3_out
0	Day 01-07	1.6219	0.0411	0.2568	2.0584	0.0303	0.2536
1	Day 08-14	2.0638	0.0637	0.7215	2.9953	0.0487	0.6841
2	Day 15-20	1.8994	0.0703	0.7465	2.5381	0.0583	0.6190
3	Day 20-27	1.7941	0.0848	0.7322	2.7897	0.0809	0.6472
4	Day 28-60	1.2144	0.1208	0.6822	2.2804	0.1371	0.6356

```
# Actual value plot
sns.lineplot(x="days_prior", y="cxl_rate",hue = 'product_type_regroup3',
data=train_nyc);
```




```
# Add prediction to train_nyc
train_nyc['g3_survive_pred'] = model_g3.predict(train_nyc_X_g3).flatten()
# Predicted value plot
sns.lineplot(x="days_prior", y="g3_survive_pred",hue = 'product_type_regroup3',
data=train_nyc);
```

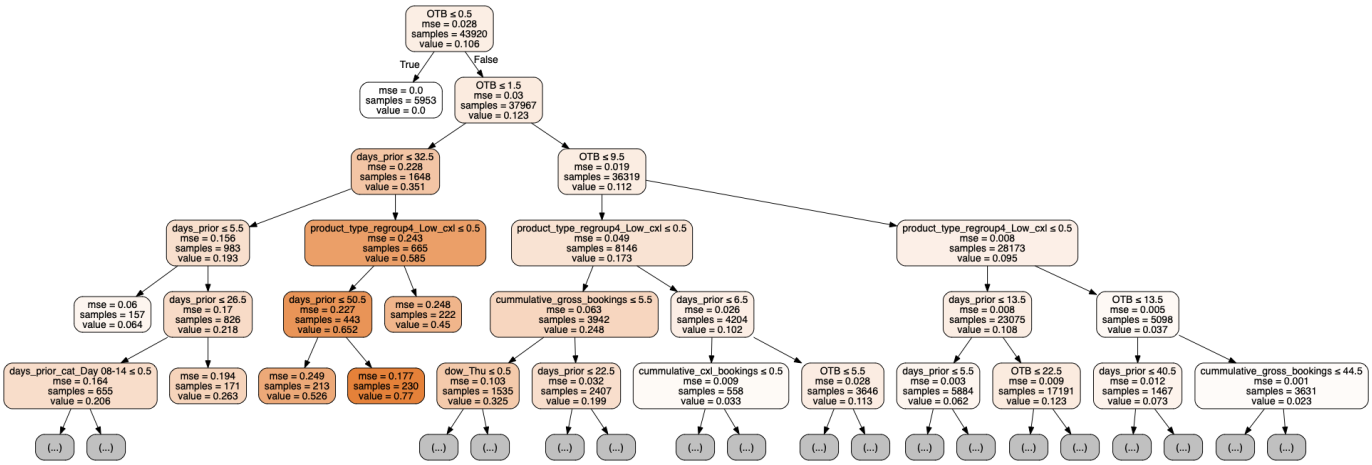


Group 4

```
model_g4 = DecisionTreeRegressor(min_samples_leaf=150,
                                 max_depth = 11)
model_g4.fit(train_nyc_X_g4, train_nyc_Y)
```

```
DecisionTreeRegressor(criterion='mse', max_depth=11, max_features=None,
                      max_leaf_nodes=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=150,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      presort=False, random_state=None, splitter='best')
```

```
visualize(model_g4, train_nyc_X_g4)
```

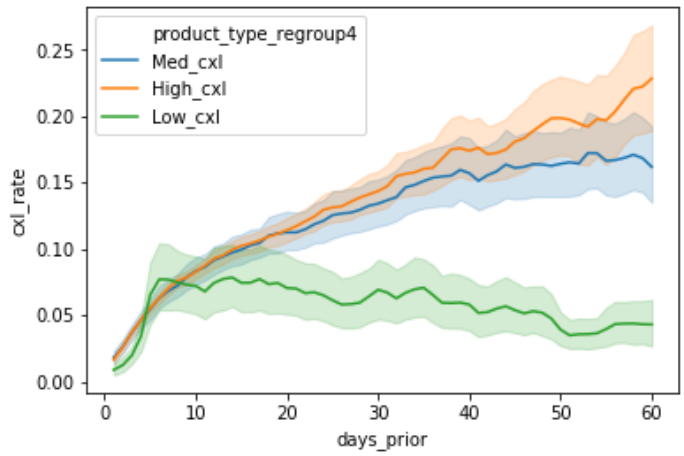



```
mae_g4_in = get_mae(train_nyc_X_g4, raw_train_nyc, model_g4, 'mae_g4_in')
mape_g4_in= get_mape(train_nyc_X_g4, raw_train_nyc, model_g4, 'mape_g4_in')
mase_g4_in = get_mase(train_nyc_X_g4, raw_train_nyc, model_g4, 'mase_g4_in')

mae_g4_out = get_mae(test_nyc_X_g4, raw_test_nyc, model_g4, 'mae_g4_out')
mape_g4_out = get_mape(test_nyc_X_g4, raw_test_nyc, model_g4, 'mape_g4_out')
mase_g4_out = get_mase(test_nyc_X_g4, raw_test_nyc, model_g4, 'mase_g4_out')

mae_g4_in.set_index('days_prior_cat').\
join(mape_g4_in.set_index('days_prior_cat')).\
join(mase_g4_in.set_index('days_prior_cat')).\
join(mae_g4_out.set_index('days_prior_cat')).\
join(mape_g4_out.set_index('days_prior_cat')).\
join(mase_g4_out.set_index('days_prior_cat')).reset_index()
```

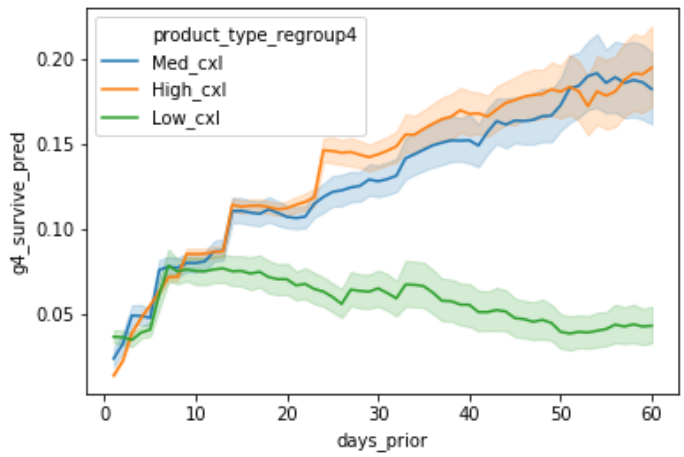
```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	days_prior_cat	mae_g4_in	mape_g4_in	mase_g4_in	mae_g4_out	mape_g4_out	mase_g4_out
0	Day 01-07	1.6856	0.0419	0.2622	2.0363	0.0343	0.2451
1	Day 08-14	2.1458	0.0637	0.6775	2.8469	0.0551	0.5971
2	Day 15-20	1.9627	0.0734	0.7747	2.6446	0.0621	0.6177
3	Day 20-27	1.7056	0.0815	0.7639	2.5221	0.0713	0.6273
4	Day 28-60	1.1917	0.1204	0.7505	2.2214	0.1117	0.6603

```
# Actual value plot
sns.lineplot(x="days_prior", y="cxl_rate",hue = 'product_type_regroup4',
data=train_nyc);
```




```
# Add prediction to train_nyc
train_nyc['g4_survive_pred'] = model_g4.predict(train_nyc_X_g4).flatten()
# Predicted value plot
sns.lineplot(x="days_prior", y="g4_survive_pred",hue = 'product_type_regroup4',
data=train_nyc);
```

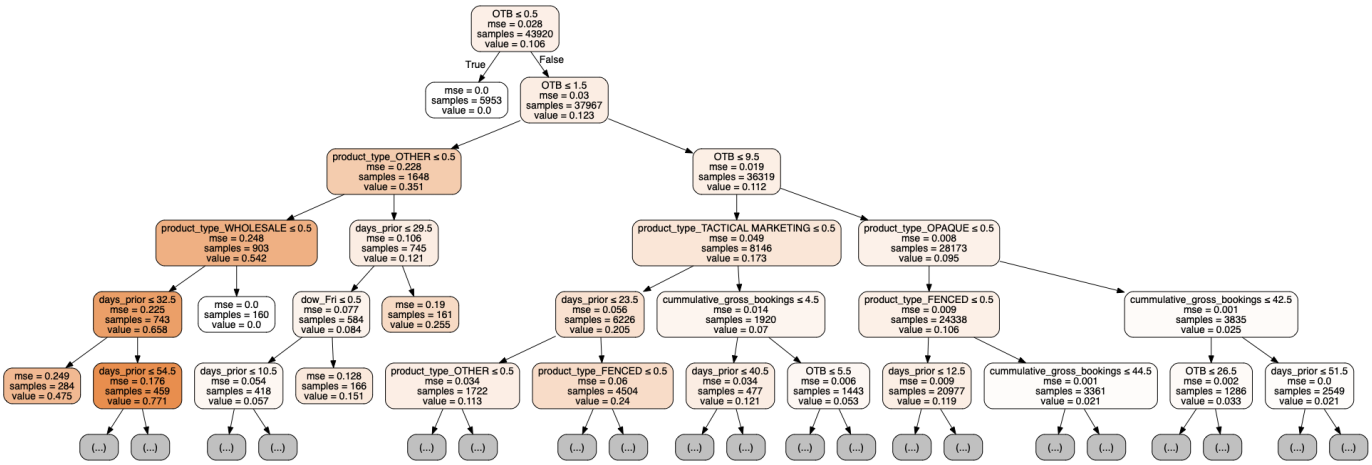


Benchmark

```
model_bm = DecisionTreeRegressor(min_samples_leaf=150,
                                 max_depth = 12)
model_bm.fit(train_nyc_X_bm, train_nyc_Y)
```

```
DecisionTreeRegressor(criterion='mse', max_depth=12, max_features=None,
                      max_leaf_nodes=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=150,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      presort=False, random_state=None, splitter='best')
```

```
visualize(model_bm, train_nyc_X_bm)
```

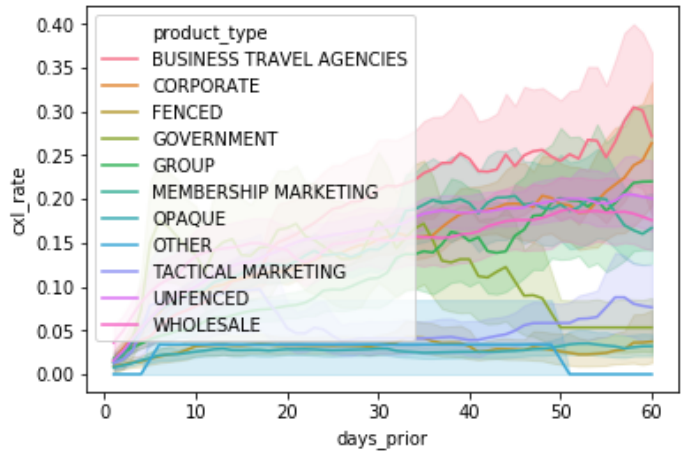



```
mae_bm_in = get_mae(train_nyc_X_bm, raw_train_nyc, model_bm, 'mae_bm_in')
mape_bm_in= get_mape(train_nyc_X_bm, raw_train_nyc, model_bm, 'mape_bm_in')
mase_bm_in = get_mase(train_nyc_X_bm, raw_train_nyc, model_bm, 'mase_bm_in')

mae_bm_out = get_mae(test_nyc_X_bm, raw_test_nyc, model_bm, 'mae_bm_out')
mape_bm_out = get_mape(test_nyc_X_bm, raw_test_nyc, model_bm, 'mape_bm_out')
mase_bm_out = get_mase(test_nyc_X_bm, raw_test_nyc, model_bm, 'mase_bm_out')

mae_bm_in.set_index('days_prior_cat').\
join(mape_bm_in.set_index('days_prior_cat')).\
join(mase_bm_in.set_index('days_prior_cat')).\
join(mae_bm_out.set_index('days_prior_cat')).\
join(mape_bm_out.set_index('days_prior_cat')).\
join(mase_bm_out.set_index('days_prior_cat')).reset_index()
```

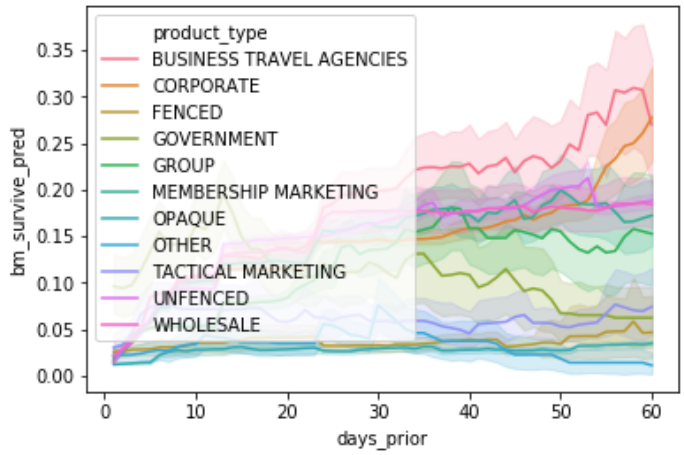
```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	days_prior_cat	mae_bm_in	mape_bm_in	mase_bm_in	mae_bm_out	mape_bm_out	mase_bm_out
0	Day 01-07	1.5330	0.0451	0.2381	1.8899	0.0308	0.2263
1	Day 08-14	1.8529	0.0623	0.5567	2.4459	0.0476	0.4718
2	Day 15-20	1.7777	0.0633	0.6365	2.5035	0.0585	0.5332
3	Day 21-27	1.5841	0.0764	0.6336	2.3105	0.0788	0.5251
4	Day 28-60	1.0869	0.1114	0.6248	2.1306	0.1470	0.6073

```
# Actual value plot
sns.lineplot(x="days_prior", y="cxl_rate",hue = 'product_type', data=train_nyc);
```




```
# Add prediction to train_nyc
train_nyc['bm_survive_pred'] = model_bm.predict(train_nyc_X_bm).flatten()
# Predicted value plot
sns.lineplot(x="days_prior", y="bm_survive_pred",hue = 'product_type',
data=train_nyc);
```




```
feature_importances = pd.DataFrame(model_bm.feature_importances_,
                                   index = train_nyc_X_bm.columns,
                                    columns=
```

```
['importance']).sort_values('importance',ascending=False)
feature_importances
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	importance
OTB	0.370138
product_type_OTHER	0.127524
days_prior	0.121396
product_type_WHOLESALE	0.108297
product_type_FENCED	0.071353
product_type_TACTICAL MARKETING	0.057386
product_type_OPAQUE	0.050276
cummulative_gross_bookings	0.027516
cummulative_cxl_bookings	0.021227
dow_Thu	0.012151
product_type_UNFENCED	0.009536
product_type_GROUP	0.008314
dow_Sun	0.005543
product_type_GOVERNMENT	0.003530
dow_Wed	0.002218
dow_Fri	0.001718
dow_Tue	0.000670
days_prior_cat_Day 01-07	0.000617
product_type_BUSINESS TRAVEL AGENCIES	0.000545
days_prior_cat_Day 28-60	0.000037
dow_Sat	0.000008
daily_gross_bookings	0.000000
daily_cxl_bookings	0.000000
dow_Mon	0.000000
daily_net_bookings	0.000000

	importance
product_type_MEMBERSHIP MARKETING	0.000000
product_type_CORPORATE	0.000000
days_prior_cat_Day 08-14	0.000000
days_prior_cat_Day 15-20	0.000000
days_prior_cat_Day 21-27	0.000000

Function to estimate best parameters

```
from sklearn.model_selection import cross_val_score, GridSearchCV

def grid_search_model(X, y):
# Perform Grid-Search
    gsc = GridSearchCV(
        estimator=DecisionTreeRegressor(),
        param_grid={
            'max_depth': range(5, 12),
            'min_samples_leaf': (50, 60, 70, 80, 90, 100, 120, 150),
        },
        cv=5, scoring='neg_mean_absolute_error', n_jobs=-1)

    grid_result = gsc.fit(X, y)
    best_params = grid_result.best_params_

    rfr = DecisionTreeRegressor(max_depth=best_params["max_depth"],\
                                min_samples_leaf=best_params["min_samples_leaf"],\
                                random_state=False)
    # Perform K-Fold CV
    scores = cross_val_score(rfr, X, y, cv=10, scoring='neg_mean_absolute_error')
    best = {'max_depth': best_params["max_depth"],
           'min_samples_leaf':best_params["min_samples_leaf"]}
    return best
```

```
grid_search_model(train_nyc_X_bm, train_nyc_Y)
```

```
{'max_depth': 11, 'min_samples_leaf': 50}
```

Metrics Summary

In-sample metrics

```
display(mae_g1_in.set_index('days_prior_cat').\
join(mae_g2_in.set_index('days_prior_cat')).\
join(mae_g3_in.set_index('days_prior_cat')).\
join(mae_g4_in.set_index('days_prior_cat')).\
```

```
join(mae_bm_in.set_index('days_prior_cat')).reset_index())

display(mape_g1_in.set_index('days_prior_cat').\
join(mape_g2_in.set_index('days_prior_cat')).\
join(mape_g3_in.set_index('days_prior_cat')).\
join(mape_g4_in.set_index('days_prior_cat')).\
join(mape_bm_in.set_index('days_prior_cat')).reset_index())

display(mase_g1_in.set_index('days_prior_cat').\
join(mase_g2_in.set_index('days_prior_cat')).\
join(mase_g3_in.set_index('days_prior_cat')).\
join(mase_g4_in.set_index('days_prior_cat')).\
join(mase_bm_in.set_index('days_prior_cat')).reset_index())
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	days_prior_cat	mae_g1_in	mae_g2_in	mae_g3_in	mae_g4_in	mae_bm_in
0	Day 01-07	1.5121	1.5551	1.6219	1.6856	1.5330
1	Day 08-14	1.8533	1.8515	2.0638	2.1458	1.8529
2	Day 15-20	1.8409	1.7488	1.8994	1.9627	1.7777
3	Day 20-27	1.7116	1.7525	1.7941	1.7056	1.5841
4	Day 28-60	1.1298	1.1848	1.2144	1.1917	1.0869

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	days_prior_cat	mape_g1_in	mape_g2_in	mape_g3_in	mape_g4_in	mape_bm_in
0	Day 01-07	0.0412	0.0396	0.0411	0.0419	0.0451
1	Day 08-14	0.0613	0.0590	0.0637	0.0637	0.0623
2	Day 15-20	0.0689	0.0638	0.0703	0.0734	0.0633
3	Day 20-27	0.0805	0.0812	0.0848	0.0815	0.0764
4	Day 28-60	0.1171	0.1153	0.1208	0.1204	0.1114

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	days_prior_cat	mase_g1_in	mase_g2_in	mase_g3_in	mase_g4_in	mase_bm_in
0	Day 01-07	0.2310	0.2444	0.2568	0.2622	0.2381
1	Day 08-14	0.5759	0.5754	0.7215	0.6775	0.5567
2	Day 15-20	0.7030	0.6480	0.7465	0.7747	0.6365
3	Day 20-27	0.7133	0.7295	0.7322	0.7639	0.6336
4	Day 28-60	0.6459	0.6613	0.6822	0.7505	0.6248

Out-sample metrics

```
display(mae_g1_out.set_index('days_prior_cat').\
join(mae_g2_out.set_index('days_prior_cat')).\
join(mae_g3_out.set_index('days_prior_cat')).\
join(mae_g4_out.set_index('days_prior_cat')).\
join(mae_bm_out.set_index('days_prior_cat')).reset_index())

display(mape_g1_out.set_index('days_prior_cat').\
join(mape_g2_out.set_index('days_prior_cat')).\
join(mape_g3_out.set_index('days_prior_cat')).\
join(mape_g4_out.set_index('days_prior_cat')).\
join(mape_bm_out.set_index('days_prior_cat')).reset_index())

display(mase_g1_out.set_index('days_prior_cat').\
join(mase_g2_out.set_index('days_prior_cat')).\
join(mase_g3_out.set_index('days_prior_cat')).\
join(mase_g4_out.set_index('days_prior_cat')).\
join(mase_bm_out.set_index('days_prior_cat')).reset_index())
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	days_prior_cat	mae_g1_out	mae_g2_out	mae_g3_out	mae_g4_out	mae_bm_out
0	Day 01-07	1.8497	2.0244	2.0584	2.0363	1.8899
1	Day 08-14	2.3671	2.6219	2.9953	2.8469	2.4459

	days_prior_cat	mae_g1_out	mae_g2_out	mae_g3_out	mae_g4_out	mae_bm_out
2	Day 15-20	2.3615	2.4186	2.5381	2.6446	2.5035
3	Day 20-27	2.6040	2.6999	2.7897	2.5221	2.3105
4	Day 28-60	2.1969	2.2721	2.2804	2.2214	2.1306

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	days_prior_cat	mape_g1_out	mape_g2_out	mape_g3_out	mape_g4_out	mape_bm_out
0	Day 01-07	0.0304	0.0311	0.0303	0.0343	0.0308
1	Day 08-14	0.0462	0.0478	0.0487	0.0551	0.0476
2	Day 15-20	0.0543	0.0564	0.0583	0.0621	0.0585
3	Day 20-27	0.0792	0.0772	0.0809	0.0713	0.0788
4	Day 28-60	0.1338	0.1560	0.1371	0.1117	0.1470

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	days_prior_cat	mase_g1_out	mase_g2_out	mase_g3_out	mase_g4_out	mase_bm_out
0	Day 01-07	0.2176	0.2459	0.2536	0.2451	0.2263
1	Day 08-14	0.4774	0.5470	0.6841	0.5971	0.4718
2	Day 15-20	0.5614	0.5766	0.6190	0.6177	0.5332
3	Day 20-27	0.6190	0.6467	0.6472	0.6273	0.5251
4	Day 28-60	0.6084	0.6305	0.6356	0.6603	0.6073

Add prediction of model bm to treated data

```
train_tree_pred = pd.DataFrame()
test_tree_pred = pd.DataFrame()

train_tree_pred['tree_pred'] = model_bm.predict(train_nyc_X_bm).flatten()
test_tree_pred['tree_pred'] = model_bm.predict(test_nyc_X_bm).flatten()
```

```
train_tree_pred.to_csv('../treated data/train_nyc_tree_pred.csv')
test_tree_pred.to_csv('../treated data/test_nyc_tree_pred.csv')
```

```
train_tree_pred.shape
```

```
(43920, 1)
```