Random forests is an ensemble learning method for regression and classification. As we mentioned in above, decision trees that are grown very deep tend to learn highly irregular patterns: they overfit their training datasets. Random forests are a way of averaging multiple deep decision trees, trained on different parts of the same training set, with the goal of reducing the variance. This comes at the expense of a small increase in the bias and some loss of interpretability, but generally greatly boosts the performance in the final model.

Random forests surely do a good job at classification but not as for regression problem as random forests don't gives precise continuous nature prediction. In case of regression, it doesn't predict beyond the range in the training data, and that they may over fit data sets that are particularly noisy. Another disadvantage of random forests is that it can feel like a black box approach for a statistical modelers. We have very little control on what the model does.

OTB, Days Prior are the 2 most important factors in our random forest. This score of the feature importance in random forest is measured by the mean decreased of impurity (variance), weighted by the probability of reaching that node, and averaged over all trees of the forest. Important features in this random forest were quite similar to decision tree's, with Days Prior became the second important feature. The pattern of average cancellation trend in random forest resembles the actual cancellation rate trend more than the Tree's pattern.

Load packages

```
# 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.ensemble import
RandomForestRegressor,AdaBoostRegressor,GradientBoostingRegressor
```

Import data

```
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

```
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
```

Group 1

```
model_g1 = RandomForestRegressor(n_estimators = 100,
                                 min_samples_split=200,
                                 max_depth = 11)
model_g1.fit(train_nyc_X_g1, train_nyc_Y)

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
0	Day 01-07	1.4849	0.0384	0.2315	1.8836	0.0296	0.2277
1	Day 08-14	1.8092	0.0591	0.5790	2.4670	0.0468	0.5208
2	Day 15-20	1.7974	0.0640	0.6826	2.3199	0.0554	0.5402
3	Day 20-27	1.6197	0.0760	0.6854	2.5186	0.0770	0.6055
4	Day 28-60	1.0540	0.1050	0.6040	2.1616	0.1504	0.6037

```
from sklearn.ensemble import AdaBoostRegressor

model_g1_rf = RandomForestRegressor(#n_estimators = 100,
                                    #min_samples_split=200,
                                    max_depth = 11)


model_g1 = AdaBoostRegressor(model_g1_rf,
                             n_estimators=300, loss='square')

model_g1.fit(train_nyc_X_g1, train_nyc_Y)
```

```
/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-
packages/sklearn/ensemble/forest.py:245: FutureWarning: The default value of
n_estimators will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-
packages/sklearn/ensemble/forest.py:245: FutureWarning: The default value of
n_estimators will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-
packages/sklearn/ensemble/forest.py:245: FutureWarning: The default value of
n_estimators will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-
packages/sklearn/ensemble/forest.py:245: FutureWarning: The default value of
n_estimators will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-
packages/sklearn/ensemble/forest.py:245: FutureWarning: The default value of
n_estimators will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-
packages/sklearn/ensemble/forest.py:245: FutureWarning: The default value of
n_estimators will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-
packages/sklearn/ensemble/forest.py:245: FutureWarning: The default value of
n_estimators will change from 10 in version 0.20 to 100 in 0.22.
```

```
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-
packages/sklearn/ensemble/forest.py:245: FutureWarning: The default value of
n_estimators will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-
packages/sklearn/ensemble/forest.py:245: FutureWarning: The default value of
n_estimators will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-
packages/sklearn/ensemble/forest.py:245: FutureWarning: The default value of
n_estimators will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-
packages/sklearn/ensemble/forest.py:245: FutureWarning: The default value of
n_estimators will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-
packages/sklearn/ensemble/forest.py:245: FutureWarning: The default value of
n_estimators will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-
packages/sklearn/ensemble/forest.py:245: FutureWarning: The default value of
n_estimators will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-
packages/sklearn/ensemble/forest.py:245: FutureWarning: The default value of
n_estimators will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-
packages/sklearn/ensemble/forest.py:245: FutureWarning: The default value of
n_estimators will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-
packages/sklearn/ensemble/forest.py:245: FutureWarning: The default value of
n_estimators will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-
packages/sklearn/ensemble/forest.py:245: FutureWarning: The default value of
n_estimators will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

```
AdaBoostRegressor(base_estimator=RandomForestRegressor(bootstrap=True,
                                                       criterion='mse',
                                                       max_depth=11,
                                                       max_features='auto',
                                                       max_leaf_nodes=None,
                                                       min_impurity_decrease=0.0,
                                                       min_impurity_split=None,
                                                       min_samples_leaf=1,
                                                       min_samples_split=2,
                                                       min_weight_fraction_leaf=0.0,
                                                       n_estimators='warn',
                                                       n_jobs=None,
```

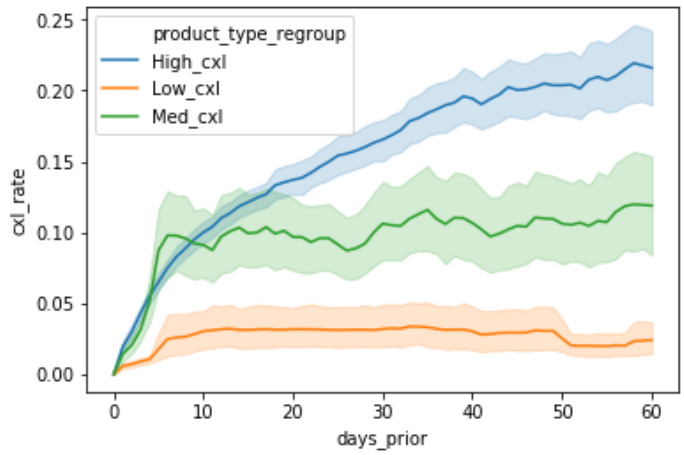
```
                                                    oob_score=False,
                                                    random_state=None,
                                                    verbose=0,
                                                    warm_start=False),
                  learning_rate=1.0, loss='square', n_estimators=300,
                  random_state=None)
```

```
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()
```

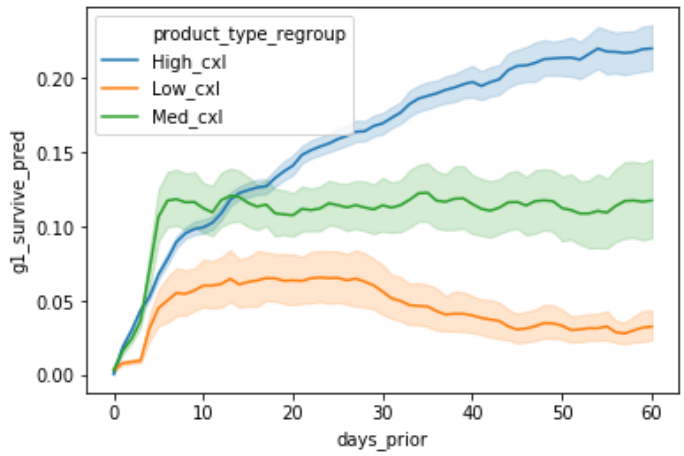
```
.dataframe tbody tr th {
    vertical-align: top;
}

.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
0	Day 01-07	1.3987	0.0386	0.2168	1.9074	0.0344	0.2308
1	Day 08-14	1.6917	0.0670	0.5378	2.4836	0.0549	0.5257
2	Day 15-20	1.6829	0.0714	0.6459	2.3670	0.0608	0.5604
3	Day 20-27	1.5163	0.0855	0.6388	2.5112	0.0810	0.6106
4	Day 28-60	0.9413	0.0946	0.5247	2.1482	0.1383	0.5893

```
# 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);
```



Benchmark

```
model_bm = RandomForestRegressor(n_estimators = 100,
                                 min_samples_split=200,
                                 max_depth = 12)
model_bm.fit(train_nyc_X_bm, train_nyc_Y)

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	1.4914	0.0390	0.2330	1.9169	0.0304	0.2315
1	Day 08-14	1.7807	0.0548	0.5642	2.4348	0.0469	0.5054
2	Day 15-20	1.7308	0.0582	0.6423	2.4466	0.0584	0.5557
3	Day 21-27	1.5191	0.0698	0.6338	2.4284	0.0735	0.5838
4	Day 28-60	1.0339	0.1013	0.5879	2.0879	0.1380	0.6038

```
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	0.337714
days_prior	0.123679
product_type_OTHER	0.097886
product_type_WHOLESALE	0.091720
product_type_FENCED	0.074174
cummulative_gross_bookings	0.061492
product_type_TACTICAL MARKETING	0.048337
product_type_OPAQUE	0.044581
product_type_BUSINESS TRAVEL AGENCIES	0.022553

	importance
dow_Thu	0.015723
cummulative_cxl_bookings	0.014517
product_type_UNFENCED	0.013284
dow_Sat	0.009909
product_type_GROUP	0.008489
product_type_GOVERNMENT	0.008184
dow_Fri	0.007876
dow_Mon	0.004661
dow_Sun	0.004335
dow_Tue	0.004178
product_type_MEMBERSHIP MARKETING	0.001804
dow_Wed	0.001452
days_prior_cat_Day 28-60	0.000957
daily_net_bookings	0.000576
product_type_CORPORATE	0.000540
days_prior_cat_Day 01-07	0.000522
daily_cxl_bookings	0.000436
daily_gross_bookings	0.000141
days_prior_cat_Day 21-27	0.000118
days_prior_cat_Day 15-20	0.000115
days_prior_cat_Day 08-14	0.000047

Model 2

```
model_g2 = RandomForestRegressor(n_estimators = 100,
                                 min_samples_split=200,
                                 max_depth = 12)
model_g2.fit(train_nyc_X_g2, train_nyc_Y)

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
0	Day 01-07	1.4814	0.0373	0.2323	2.0187	0.0302	0.2463
1	Day 08-14	1.7340	0.0575	0.5619	2.8281	0.0473	0.6277
2	Day 15-20	1.7208	0.0614	0.6849	2.5820	0.0546	0.6580
3	Day 20-27	1.6428	0.0760	0.7138	2.6360	0.0721	0.6451
4	Day 28-60	1.0492	0.1060	0.6112	2.2279	0.1289	0.6113

Model 3

```
model_g3 = RandomForestRegressor(n_estimators = 150,
                                 min_samples_split=200,
                                 max_depth = 15)
model_g3.fit(train_nyc_X_g3, train_nyc_Y)

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_in	mape_g3_in	mase_g3_in	mae_g3_out	mape_g3_out	mase_g3_out
0	Day 01-07	1.4781	0.0378	0.2303	1.9596	0.0300	0.2377
1	Day 08-14	1.7545	0.0593	0.5856	2.7000	0.0478	0.6022
2	Day 15-20	1.7788	0.0645	0.7288	2.4815	0.0575	0.6238
3	Day 20-27	1.5813	0.0780	0.6820	2.6738	0.0783	0.6456
4	Day 28-60	1.0174	0.1088	0.5906	2.3208	0.1341	0.6285

Model 4

```
model_g4 = RandomForestRegressor(n_estimators = 100,
                                 min_samples_split=200,
                                 max_depth = 12)
model_g4.fit(train_nyc_X_g4, train_nyc_Y)

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	1.5978	0.0392	0.2499	2.0433	0.0336	0.2532
1	Day 08-14	1.9318	0.0594	0.6357	2.6602	0.0527	0.6059
2	Day 15-20	1.8745	0.0692	0.7844	2.4284	0.0593	0.5981
3	Day 20-27	1.6525	0.0771	0.7489	2.4657	0.0714	0.6170
4	Day 28-60	1.1331	0.1093	0.7270	2.1565	0.1118	0.6486

Summary

```
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	1.3987	1.4814	1.4781	1.5978	1.4848
1	Day 08-14	1.6917	1.7340	1.7545	1.9318	1.7740
2	Day 15-20	1.6829	1.7208	1.7788	1.8745	1.7313
3	Day 20-27	1.5163	1.6428	1.5813	1.6525	1.5155
4	Day 28-60	0.9413	1.0492	1.0174	1.1331	1.0418

```
.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	0.0386	0.0373	0.0378	0.0392	0.0388
1	Day 08-14	0.0670	0.0575	0.0593	0.0594	0.0551

	days_prior_cat	mape_g1_in	mape_g2_in	mape_g3_in	mape_g4_in	mape_bm_in
2	Day 15-20	0.0714	0.0614	0.0645	0.0692	0.0587
3	Day 20-27	0.0855	0.0760	0.0780	0.0771	0.0700
4	Day 28-60	0.0946	0.1060	0.1088	0.1093	0.1015

```
.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	0.2168	0.2323	0.2303	0.2499	0.2312
1	Day 08-14	0.5378	0.5619	0.5856	0.6357	0.5597
2	Day 15-20	0.6459	0.6849	0.7288	0.7844	0.6409
3	Day 20-27	0.6388	0.7138	0.6820	0.7489	0.6316
4	Day 28-60	0.5247	0.6112	0.5906	0.7270	0.5874

```
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	1.9074	2.0187	1.9596	2.0433	1.9108
1	Day 08-14	2.4836	2.8281	2.7000	2.6602	2.4198
2	Day 15-20	2.3670	2.5820	2.4815	2.4284	2.4620
3	Day 20-27	2.5112	2.6360	2.6738	2.4657	2.4036
4	Day 28-60	2.1482	2.2279	2.3208	2.1565	2.1001

```
.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	0.0344	0.0302	0.0300	0.0336	0.0300
1	Day 08-14	0.0549	0.0473	0.0478	0.0527	0.0460
2	Day 15-20	0.0608	0.0546	0.0575	0.0593	0.0591
3	Day 20-27	0.0810	0.0721	0.0783	0.0714	0.0730
4	Day 28-60	0.1383	0.1289	0.1341	0.1118	0.1425

```
.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	0.2308	0.2463	0.2377	0.2532	0.2301
1	Day 08-14	0.5257	0.6277	0.6022	0.6059	0.5011
2	Day 15-20	0.5604	0.6580	0.6238	0.5981	0.5564
3	Day 20-27	0.6106	0.6451	0.6456	0.6170	0.5738
4	Day 28-60	0.5893	0.6113	0.6285	0.6486	0.6009

Export

```
train_rf_pred = pd.DataFrame()
test_rf_pred = pd.DataFrame()

train_rf_pred['rf_pred'] = model_bm.predict(train_nyc_X_bm).flatten()
test_rf_pred['rf_pred'] = model_bm.predict(test_nyc_X_bm).flatten()
```

```
train_rf_pred.to_csv('../treated data/train_rf_pred.csv')
test_rf_pred.to_csv('../treated data/test_rf_pred.csv')
```

To find parameter

```
from sklearn.ensemble import RandomForestRegressor # To run random forest
from sklearn.model_selection import cross_val_score, GridSearchCV

def rfr_model(X, y):
# Perform Grid-Search
    gsc = GridSearchCV(
        estimator=RandomForestRegressor(),
        param_grid={
            'max_depth': range(3,7),
            'n_estimators': (10, 50, 100, 1000),
        },
        cv=5, scoring='neg_mean_squared_error', verbose=0, n_jobs=-1)

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

    rfr = RandomForestRegressor(max_depth=best_params["max_depth"],
n_estimators=best_params["n_estimators"],
random_state=False, verbose=False)
# Perform K-Fold CV
    scores = cross_val_score(rfr, X, y, cv=10, scoring='neg_mean_absolute_error')

    return scores
```

```
rfr_model(train_nyc_X_g1, train_nyc_Y)
```