round_7

April 17, 2025

[7]:
```
from src.future_forecast import forecast_future
from src.preprocessing import load_data, split_processed_data
from src.forecasting import process_portfolio, rolling_predict
from src.feature_engineering import create_features
from src.model_fit import train_models
import numpy as np
import joblib
import os
os.makedirs('models/round7_full', exist_ok=True)
os.makedirs('models/round7', exist_ok=True)
os.makedirs('models/round7_simple', exist_ok=True)
import warnings
warnings.filterwarnings("ignore")
```

[8]:
```
data_dict = load_data('data/round_7/*.xlsx')
```

[9]:
```
data_dict
```

[9]:
```
{'Hong Kong International Airport':            Date  Total
 0    2023-01-08  30275
 1    2023-01-09  24796
 2    2023-01-10  24012
 3    2023-01-11  23273
 4    2023-01-12  23053
 ..          …      …
 823 2025-04-10  55832
 824 2025-04-11  64252
 825 2025-04-12  62073
 826 2025-04-13  68078
 827 2025-04-14  64296

 [828 rows x 2 columns],
 'Hong Kong-Zhuhai-Macao Bridge':           Date  Total
 0    2023-01-08   5321
 1    2023-01-09   4937
 2    2023-01-10   5617
 3    2023-01-11   6191
 4    2023-01-12   6699
```

```
..          …     …
823 2025-04-10  28153
824 2025-04-11  31023
825 2025-04-12  34587
826 2025-04-13  55886
827 2025-04-14  36194

[828 rows x 2 columns],
'Lo Wu':          Date   Total
0   2023-02-06   30319
1   2023-02-07   32954
2   2023-02-08   35022
3   2023-02-09   34764
4   2023-02-10   39235
..          …      …
794 2025-04-10   80281
795 2025-04-11   82676
796 2025-04-12   81797
797 2025-04-13  119651
798 2025-04-14   87925

[799 rows x 2 columns],
'Lok Ma Chau Spur Line':          Date  Total
0   2023-01-08   5680
1   2023-01-09   6455
2   2023-01-10   7910
3   2023-01-11   8594
4   2023-01-12  10113
..          …     …
823 2025-04-10  65681
824 2025-04-11  68887
825 2025-04-12  68802
826 2025-04-13  99840
827 2025-04-14  71513

[828 rows x 2 columns],
'Shenzhen Bay':          Date  Total
0   2023-01-08   3543
1   2023-01-09   3265
2   2023-01-10   3930
3   2023-01-11   4271
4   2023-01-12   5030
..          …     …
823 2025-04-10  50984
824 2025-04-11  53957
825 2025-04-12  51832
826 2025-04-13  88373
```

```
827 2025-04-14  55989

[828 rows x 2 columns]}
```

[10]:
```python
processed_data_dict = {}
df_dict = {}
for name, data in data_dict.items():
    df ,processed_data = create_features(data)
    df_dict[name] = df
    processed_data_dict[name] = processed_data
```

[11]:
```python
processed_data_dict['Lo Wu'].tail()
```

[11]:
```
           Date   Total  airport_sum  airport_pc  airport_mobile  bay_sum  \
794 2025-04-10   80281          263          89             174      221
795 2025-04-11   82676          272          91             181      254
796 2025-04-12   81797          257          76             181      234
797 2025-04-13  119651          256          76             180      233
798 2025-04-14   87925          280         116             164      306

     bay_pc  bay_mobile  bridge_sum  bridge_pc  …  ex_rate_volatility_7d  \
794      85         136        1533        347  …               0.001380
795      87         167        1650        341  …               0.001536
796      74         160        1516        279  …               0.001561
797      68         165        1410        234  …               0.002241
798     127         179        1546        391  …               0.003553

     rolling_7_mean  rolling_7_std  rolling_30_mean  ma_ratio_7_30  \
794   101769.142857   24776.174963     92987.233333       1.094442
795   102405.428571   24045.320285     92956.000000       1.101655
796    95880.428571   21935.940318     93117.033333       1.029677
797    91276.285714   20847.890155     93166.200000       0.979715
798    88708.142857   14249.459831     94429.033333       0.939416

     rolling_7_ex_rate_mean  month_sin  days_since_start  days_squared  \
794                0.934414   0.866025               794        630436
795                0.934814   0.866025               795        632025
796                0.935214   0.866025               796        633616
797                0.936371   0.866025               797        635209
798                0.937814   0.866025               798        636804

     weekly_growth
794       0.952972
795       1.565046
796       1.144196
797       1.062043
798       1.119626
```

```
[5 rows x 136 columns]
```

[14]:
```python
df_dict['Lo Wu'].tail()
```

[14]:
```
           Date   Total      ex  airport_sum  airport_pc  airport_mobile  \
794 2025-04-10   80281  0.9428          263          89             174
795 2025-04-11   82676  0.9429          272          91             181
796 2025-04-12   81797  0.9429          257          76             181
797 2025-04-13  119651  0.9429          256          76             180
798 2025-04-14   87925  0.9429          280         116             164

     bay_sum  bay_pc  bay_mobile  bridge_sum  …  ex_rate_volatility_7d  \
794      221      85         136        1533  …               0.001380
795      254      87         167        1650  …               0.001536
796      234      74         160        1516  …               0.001561
797      233      68         165        1410  …               0.002241
798      306     127         179        1546  …               0.003553

     rolling_7_mean  rolling_7_std  rolling_30_mean  ma_ratio_7_30  \
794   101769.142857   24776.174963     92987.233333       1.094442
795   102405.428571   24045.320285     92956.000000       1.101655
796    95880.428571   21935.940318     93117.033333       1.029677
797    91276.285714   20847.890155     93166.200000       0.979715
798    88708.142857   14249.459831     94429.033333       0.939416

     rolling_7_ex_rate_mean  month_sin  days_since_start  days_squared  \
794                0.934414   0.866025               794        630436
795                0.934814   0.866025               795        632025
796                0.935214   0.866025               796        633616
797                0.936371   0.866025               797        635209
798                0.937814   0.866025               798        636804

     weekly_growth
794       0.952972
795       1.565046
796       1.144196
797       1.062043
798       1.119626

[5 rows x 165 columns]
```

[15]:
```python
full_models = []
for name, processed_data in list(processed_data_dict.items()):
    X_train, X_test, y_train, y_test = split_processed_data(processed_data)

    #
```

```
    best_models, stacking, mape = train_models(X_train, y_train, X_test, y_test)

    #
    best_model_name = min(mape, key=mape.get)
    if best_model_name == 'stacking':
        best_model = stacking
    else:
        best_model = best_models[best_model_name]

    #
    filename = f"models/round7_full/{name}_best_{best_model_name}.pkl"
    joblib.dump({
        'model': best_model,
        'features': X_train.columns.tolist(),
        'name': name,
        'mape': mape[best_model_name]
    },
    filename)
    full_models.append(filename)
    stacking_filename = f"models/round7_full/{name}_stacking.pkl"
    joblib.dump(stacking, stacking_filename)
    print(f"{name}     {filename} (MAPE={mape[best_model_name]:.4f})")
```

```
Hong Kong International Airport     models/round7_full/Hong Kong
International Airport_best_catboost.pkl (MAPE=0.0485)
Hong Kong-Zhuhai-Macao Bridge     models/round7_full/Hong Kong-Zhuhai-Macao
Bridge_best_catboost.pkl (MAPE=0.0804)
Lo Wu     models/round7_full/Lo Wu_best_catboost.pkl (MAPE=0.0751)
Lok Ma Chau Spur Line     models/round7_full/Lok Ma Chau Spur
Line_best_catboost.pkl (MAPE=0.0767)
Shenzhen Bay     models/round7_full/Shenzhen Bay_best_catboost.pkl
(MAPE=0.0955)
```

[16]:

```
models = []
for name, processed_data in list(processed_data_dict.items()):
    X_train, X_test, y_train, y_test = split_processed_data(processed_data)

    #
    best_models, stacking, mape = train_models(X_train, y_train, X_test, y_test)

    #
    best_model_name = min(mape, key=mape.get)
    if best_model_name == 'stacking':
        best_model = stacking
    else:
        best_model = best_models[best_model_name]
```

```
    #
    if best_model_name == 'stacking':
        #
        base_model = best_model.estimators_[0]
        importance = base_model.feature_importances_
        features = base_model.feature_names_in_
    elif hasattr(best_model, 'feature_importances_'):
        importance = best_model.feature_importances_
        #
        if hasattr(best_model, 'feature_names_in_'):  # sklearn
            features = best_model.feature_names_in_
        elif hasattr(best_model, 'feature_name_'):  # LightGBM
            features = best_model.feature_name_
        elif hasattr(best_model, 'feature_names_'):  # CatBoost
            features = best_model.feature_names_
        else:
            features = X_train.columns.tolist()
    elif hasattr(best_model, 'get_feature_importance'):
        importance = best_model.get_feature_importance()
        features = best_model.feature_names_
    else:
        features = X_train.columns.tolist()
        importance = np.ones(len(features))

    #
    sorted_idx = np.argsort(importance)[::-1]
    cumulative = np.cumsum(importance[sorted_idx])
    threshold_idx = np.where(cumulative >= 0.9 * cumulative[-1])[0][0]
    min_features = 20  #
    selected_features = [features[i] for i in sorted_idx[:max(threshold_idx+1, min_features)]]
    print(f'{name} selected features:', selected_features)
    #
    if len(selected_features) < len(features):
        print(f"{name}    :   {len(features)}     {len(selected_features)}  ")
        X_train_selected = X_train[selected_features]
        X_test_selected = X_test[selected_features]

        #
        best_models, stacking, mape = train_models(X_train_selected, y_train, X_test_selected, y_test)
        best_model_name = min(mape, key=mape.get)
        if best_model_name == 'stacking':
            best_model = stacking
        else:
            best_model = best_models[best_model_name]
```

```
    #
    filename = f"models/round7/{name}_best_{best_model_name}.pkl"
    #
    joblib.dump({
        'model': best_model,
        'features': selected_features if 'selected_features' in locals() else␣
↪X_train.columns.tolist(),
        'name': name,
        'mape': mape[best_model_name]
    },
        filename)
    models.append(filename)
    stacking_filename = f"models/round7/{name}_stacking.pkl"
    joblib.dump(stacking, stacking_filename)
    print(f"{name}     {filename} (MAPE={mape[best_model_name]:.4f})")
```

```
Hong Kong International Airport selected features: ['lag_1', 'rolling_7_mean',
'is_hk_holiday', 'days_since_start', 'ma_ratio_7_30', 'weekday', 'lag_7',
'rolling_30_mean', 'post_cn_holiday', 'lag_21', 'airport_pc', 'ex_rate_lag_5',
'ex_rate_lag_7', 'lowu_pc', 'year', 'bay_sum', 'lag_4', 'lag_2',
'hk_weather_mobile_lag_7', 'month_sin', 'lowu_sum', 'days_squared',
'hk_map_pc_lag_7', 'lowu_mobile', 'rolling_7_std', 'bridge_pc',
'hk_show_mobile_lag_5', 'day', 'hk_weather_sum_lag_7', 'lag_5', 'hk_pc_lag_7',
'hk_map_sum_lag_14', 'hk_hotel_sum_lag_7', 'hk_weekend_holiday',
'hk_shopping_pc_lag_14', 'bay_mobile', 'post_hk_holiday', 'hk_food_pc_lag_5',
'pc_mobile_lag_14', 'bridge_mobile', 'hk_map_sum_lag_5', 'hk_show_pc_lag_5',
'hk_pc_lag_5', 'hk_weather_sum_lag_5', 'month', 'airport_mobile', 'rain',
'hk_weather_pc_lag_14', 'hk_food_sum_lag_7', 'bay_pc', 'total_change_7d',
'hk_shopping_sum_lag_5']
Hong Kong International Airport   :   134     52
Hong Kong International Airport     models/round7/Hong Kong International
Airport_best_catboost.pkl (MAPE=0.0474)
Hong Kong-Zhuhai-Macao Bridge selected features: ['is_hk_holiday', 'lag_1',
'airport_pc', 'bridge_pc', 'is_cn_holiday', 'bay_pc', 'days_squared', 'weekday',
'hk_map_pc_lag_7', 'year', 'lag_2', 'pc_mobile_lag_7', 'days_since_start',
'baidu_mobile_lag_7', 'rain', 'rolling_7_mean', 'is_weekend', 'lag_14',
'bridge_mobile', 'hk_weather_sum_lag_7', 'month_sin', 'bay_sum',
'rolling_30_mean', 'ex_rate_lag_7', 'hk_show_pc_lag_5', 'ma_ratio_7_30',
'month', 'pc_mobile_lag_14', 'hk_pc_lag_7', 'day', 'lag_21', 'post_hk_holiday',
'hk_pc_lag_5', 'lowu_pc', 'total_change_14d', 'hk_weather_sum_lag_5', 'quarter',
'cn_weekend_holiday']
Hong Kong-Zhuhai-Macao Bridge   :   134     38
Hong Kong-Zhuhai-Macao Bridge     models/round7/Hong Kong-Zhuhai-Macao
Bridge_best_catboost.pkl (MAPE=0.0737)
Lo Wu selected features: ['is_hk_holiday', 'lag_1', 'days_since_start',
'bay_pc', 'is_cn_holiday', 'weekday', 'days_squared', 'lag_14', 'lag_2',
'airport_pc', 'lag_21', 'rolling_30_mean', 'rain', 'bay_mobile',
'hk_map_sum_lag_7', 'hk_pc_lag_7', 'bridge_pc', 'hk_metro_mobile_lag_14',
```

```
'hk_sum_lag_7', 'lowu_mobile', 'bay_sum', 'hk_show_sum_lag_5', 'bridge_mobile',
'month_sin', 'day', 'airport_sum', 'bridge_sum', 'lag_7', 'hk_map_pc_lag_7',
'quarter', 'ex_rate_lag_5', 'hk_mobile_lag_7', 'lowu_pc', 'pc_mobile_lag_7',
'ex_rate_lag_7', 'hk_metro_pc_lag_14', 'hk_pc_lag_14', 'hk_show_pc_lag_14',
'hk_weather_sum_lag_7']
Lo Wu   :  134     39
Lo Wu     models/round7/Lo Wu_best_stacking.pkl (MAPE=0.0729)
Lok Ma Chau Spur Line selected features: ['is_hk_holiday', 'lag_1',
'is_cn_holiday', 'bay_pc', 'days_since_start', 'bridge_pc', 'airport_pc',
'lag_14', 'lag_21', 'weekday', 'rain', 'total_change_7d', 'bay_sum',
'rolling_7_mean', 'hk_pc_lag_14', 'lowu_sum', 'days_squared', 'ex_rate_lag_5',
'hk_map_sum_lag_7', 'is_weekend', 'lag_2', 'airport_mobile',
'hk_show_sum_lag_7', 'bay_mobile', 'lag_7', 'hk_pc_lag_7', 'hk_hotel_sum_lag_7',
'bridge_mobile', 'rolling_30_mean', 'hk_map_pc_lag_14', 'ex_rate_lag_7',
'lowu_mobile', 'day', 'ma_ratio_7_30', 'post_cn_holiday', 'month_sin',
'hk_show_sum_lag_5', 'hk_hotel_sum_lag_5', 'cn_weekend_holiday',
'hk_map_pc_lag_7', 'lag_4', 'month', 'ex_rate_lag_14', 'hk_food_mobile_lag_14',
'hk_weather_pc_lag_14', 'rolling_7_std']
Lok Ma Chau Spur Line   :  134     46
Lok Ma Chau Spur Line     models/round7/Lok Ma Chau Spur
Line_best_stacking.pkl (MAPE=0.0701)
Shenzhen Bay selected features: ['is_hk_holiday', 'lag_1', 'airport_pc',
'days_since_start', 'bridge_pc', 'days_squared', 'lag_14', 'bay_pc', 'lag_21',
'is_cn_holiday', 'weekday', 'rain', 'hk_show_sum_lag_7', 'year',
'rolling_30_mean', 'lowu_pc', 'bay_mobile', 'hk_weekend_holiday', 'bay_sum',
'pc_mobile_lag_7', 'ex_rate_lag_7', 'hk_map_pc_lag_7', 'day', 'lowu_mobile',
'lag_7', 'bridge_mobile', 'hk_pc_lag_7', 'rolling_7_ex_rate_mean',
'hk_metro_pc_lag_7', 'hk_food_pc_lag_14', 'hk_weather_sum_lag_7', 'lag_2',
'total_change_14d', 'lowu_sum', 'baidu_sum_lag_5', 'post_hk_holiday',
'hk_show_mobile_lag_5', 'airport_mobile', 'rolling_7_std', 'hk_show_sum_lag_5',
'hk_shopping_pc_lag_14', 'lag_5', 'rolling_7_mean', 'hk_map_sum_lag_5',
'hk_show_pc_lag_5', 'lag_30', 'ex_rate_volatility_7d', 'month',
'hk_hotel_sum_lag_7']
Shenzhen Bay   :  134     49
Shenzhen Bay     models/round7/Shenzhen Bay_best_catboost.pkl (MAPE=0.0916)
```

[17]:
```python
simple_models = []
for name, processed_data in list(processed_data_dict.items()):
    X_train, X_test, y_train, y_test = split_processed_data(processed_data)

    #
    best_models, stacking, mape = train_models(X_train, y_train, X_test, y_test)

    #
    best_model_name = min(mape, key=mape.get)
    if best_model_name == 'stacking':
        best_model = stacking
```

```
    else:
        best_model = best_models[best_model_name]

    #
    if best_model_name == 'stacking':
        #
        base_model = best_model.estimators_[0]
        importance = base_model.feature_importances_
        features = base_model.feature_names_in_
    elif hasattr(best_model, 'feature_importances_'):
        importance = best_model.feature_importances_
        #
        if hasattr(best_model, 'feature_names_in_'):  # sklearn
            features = best_model.feature_names_in_
        elif hasattr(best_model, 'feature_name_'):  # LightGBM
            features = best_model.feature_name_
        elif hasattr(best_model, 'feature_names_'):  # CatBoost
            features = best_model.feature_names_
        else:
            features = X_train.columns.tolist()
    elif hasattr(best_model, 'get_feature_importance'):
        importance = best_model.get_feature_importance()
        features = best_model.feature_names_
    else:
        features = X_train.columns.tolist()
        importance = np.ones(len(features))

    #
    sorted_idx = np.argsort(importance)[::-1]
    cumulative = np.cumsum(importance[sorted_idx])
    threshold_idx = np.where(cumulative >= 0.8 * cumulative[-1])[0][0]
    min_features = 20  #
    selected_features = [features[i] for i in sorted_idx[:max(threshold_idx+1, min_features)]]
    print(f'{name} selected features:', selected_features)
    #
    if len(selected_features) < len(features):
        print(f"{name}   :  {len(features)}     {len(selected_features)}  ")
        X_train_selected = X_train[selected_features]
        X_test_selected = X_test[selected_features]

        #
        best_models, stacking, mape = train_models(X_train_selected, y_train, X_test_selected, y_test)
        best_model_name = min(mape, key=mape.get)
        if best_model_name == 'stacking':
            best_model = stacking
```

```python
        else:
            best_model = best_models[best_model_name]

    #
    filename = f"models/round7_simple/{name}_best_{best_model_name}.pkl"
    #
    joblib.dump({
        'model': best_model,
        'features': selected_features if 'selected_features' in locals() else X_train.columns.tolist(),
        'name': name,
        'mape': mape[best_model_name]
    },
        filename)
    simple_models.append(filename)
    stacking_filename = f"models/round7_simple/{name}_stacking.pkl"
    joblib.dump(stacking, stacking_filename)
    print(f"{name}     {filename} (MAPE={mape[best_model_name]:.4f})")
```

```
Hong Kong International Airport selected features: ['lag_1', 'rolling_7_mean',
'is_hk_holiday', 'days_since_start', 'ma_ratio_7_30', 'weekday', 'lag_7',
'rolling_30_mean', 'post_cn_holiday', 'lag_21', 'airport_pc', 'ex_rate_lag_5',
'ex_rate_lag_7', 'lowu_pc', 'year', 'bay_sum', 'lag_4', 'lag_2',
'hk_weather_mobile_lag_7', 'month_sin', 'lowu_sum', 'days_squared',
'hk_map_pc_lag_7', 'lowu_mobile', 'rolling_7_std', 'bridge_pc',
'hk_show_mobile_lag_5', 'day', 'hk_weather_sum_lag_7']
Hong Kong International Airport   :  134     29
Hong Kong International Airport     models/round7_simple/Hong Kong
International Airport_best_catboost.pkl (MAPE=0.0458)
Hong Kong-Zhuhai-Macao Bridge selected features: ['is_hk_holiday', 'lag_1',
'airport_pc', 'bridge_pc', 'is_cn_holiday', 'bay_pc', 'days_squared', 'weekday',
'hk_map_pc_lag_7', 'year', 'lag_2', 'pc_mobile_lag_7', 'days_since_start',
'baidu_mobile_lag_7', 'rain', 'rolling_7_mean', 'is_weekend', 'lag_14',
'bridge_mobile', 'hk_weather_sum_lag_7']
Hong Kong-Zhuhai-Macao Bridge   :  134     20
Hong Kong-Zhuhai-Macao Bridge     models/round7_simple/Hong Kong-Zhuhai-
Macao Bridge_best_catboost.pkl (MAPE=0.0748)
Lo Wu selected features: ['is_hk_holiday', 'lag_1', 'days_since_start',
'bay_pc', 'is_cn_holiday', 'weekday', 'days_squared', 'lag_14', 'lag_2',
'airport_pc', 'lag_21', 'rolling_30_mean', 'rain', 'bay_mobile',
'hk_map_sum_lag_7', 'hk_pc_lag_7', 'bridge_pc', 'hk_metro_mobile_lag_14',
'hk_sum_lag_7', 'lowu_mobile']
Lo Wu   :  134     20
Lo Wu     models/round7_simple/Lo Wu_best_stacking.pkl (MAPE=0.0665)
Lok Ma Chau Spur Line selected features: ['is_hk_holiday', 'lag_1',
'is_cn_holiday', 'bay_pc', 'days_since_start', 'bridge_pc', 'airport_pc',
'lag_14', 'lag_21', 'weekday', 'rain', 'total_change_7d', 'bay_sum',
'rolling_7_mean', 'hk_pc_lag_14', 'lowu_sum', 'days_squared', 'ex_rate_lag_5',
```

```
'hk_map_sum_lag_7', 'is_weekend', 'lag_2', 'airport_mobile',
'hk_show_sum_lag_7', 'bay_mobile']
Lok Ma Chau Spur Line   :  134     24
Lok Ma Chau Spur Line     models/round7_simple/Lok Ma Chau Spur
Line_best_catboost.pkl (MAPE=0.0674)
Shenzhen Bay selected features: ['is_hk_holiday', 'lag_1', 'airport_pc',
'days_since_start', 'bridge_pc', 'days_squared', 'lag_14', 'bay_pc', 'lag_21',
'is_cn_holiday', 'weekday', 'rain', 'hk_show_sum_lag_7', 'year',
'rolling_30_mean', 'lowu_pc', 'bay_mobile', 'hk_weekend_holiday', 'bay_sum',
'pc_mobile_lag_7', 'ex_rate_lag_7', 'hk_map_pc_lag_7', 'day', 'lowu_mobile']
Shenzhen Bay   :  134     24
Shenzhen Bay     models/round7_simple/Shenzhen Bay_best_catboost.pkl
(MAPE=0.0901)
```

[18]:
```python
for (name,df), model_path in zip(df_dict.items(), full_models):
    saved_data = joblib.load(model_path)
    model = saved_data['model']
    required_features = saved_data['features']
    mape = saved_data['mape']
    print(forecast_future(df, model, 5, required_features)['Total'], name, mape)
```

```
Date
2025-04-15    59596.375592
2025-04-16    58787.574600
2025-04-17    60644.118203
2025-04-18    66813.755391
2025-04-19    69893.219826
Name: Total, dtype: float64 Hong Kong International Airport 0.04846055562850791
Date
2025-04-15    29275.936308
2025-04-16    28917.107650
2025-04-17    29748.629123
2025-04-18    47746.996536
2025-04-19    65336.235249
Name: Total, dtype: float64 Hong Kong-Zhuhai-Macao Bridge 0.08042138598519352
Date
2025-04-15     80383.841683
2025-04-16     84251.966322
2025-04-17     84593.978150
2025-04-18    117017.168128
2025-04-19    132464.546084
Name: Total, dtype: float64 Lo Wu 0.07512931748777013
Date
2025-04-15     66165.651998
2025-04-16     68474.924192
2025-04-17     67130.794542
2025-04-18     91406.381154
2025-04-19    100788.723881
```

```
Name: Total, dtype: float64 Lok Ma Chau Spur Line 0.07668981166451128
Date
2025-04-15    54038.715380
2025-04-16    53988.086592
2025-04-17    56315.624998
2025-04-18    80408.504208
2025-04-19    86172.494004
Name: Total, dtype: float64 Shenzhen Bay 0.09553796514948758
```

[19]:
```
for (name,df), model_path in zip(df_dict.items(), models):
    saved_data = joblib.load(model_path)
    model = saved_data['model']
    required_features = saved_data['features']
    mape = saved_data['mape']
    print(forecast_future(df, model, 5, required_features)['Total'], name, mape)
```

```
Date
2025-04-15    59866.717483
2025-04-16    58689.693194
2025-04-17    60073.406073
2025-04-18    67544.324198
2025-04-19    71528.703706
Name: Total, dtype: float64 Hong Kong International Airport 0.047374445032239
Date
2025-04-15    31363.481325
2025-04-16    29560.004283
2025-04-17    31908.755642
2025-04-18    49380.819498
2025-04-19    68646.891221
Name: Total, dtype: float64 Hong Kong-Zhuhai-Macao Bridge 0.07367136158537427
Date
2025-04-15     81096.948908
2025-04-16     81477.244247
2025-04-17     81124.391693
2025-04-18    109681.755307
2025-04-19    134969.488951
Name: Total, dtype: float64 Lo Wu 0.07285770778817846
Date
2025-04-15     63856.312069
2025-04-16     64704.526368
2025-04-17     64640.325405
2025-04-18    100526.221391
2025-04-19    101560.576060
Name: Total, dtype: float64 Lok Ma Chau Spur Line 0.07012339667752591
Date
2025-04-15    48160.448394
2025-04-16    47330.712695
2025-04-17    50640.493918
```

```
2025-04-18    71420.265087
2025-04-19    82567.195328
Name: Total, dtype: float64 Shenzhen Bay 0.09163044325515166
```

[20]:
```
for (name,df), model_path in zip(df_dict.items(), simple_models):
    saved_data = joblib.load(model_path)
    model = saved_data['model']
    required_features = saved_data['features']
    mape = saved_data['mape']
    print(forecast_future(df, model, 5, required_features)['Total'], name, mape)
```

```
Date
2025-04-15    56001.745192
2025-04-16    54312.258314
2025-04-17    55378.736377
2025-04-18    64060.276619
2025-04-19    65587.224745
Name: Total, dtype: float64 Hong Kong International Airport 0.045809337290135366
Date
2025-04-15    26766.810884
2025-04-16    24232.727926
2025-04-17    27329.163298
2025-04-18    45477.154122
2025-04-19    67694.471131
Name: Total, dtype: float64 Hong Kong-Zhuhai-Macao Bridge 0.07482919134370135
Date
2025-04-15     81896.228649
2025-04-16     80707.801689
2025-04-17     80447.434306
2025-04-18    112444.155212
2025-04-19    130645.709655
Name: Total, dtype: float64 Lo Wu 0.06645907966285537
Date
2025-04-15     62436.491613
2025-04-16     63631.249705
2025-04-17     65523.221383
2025-04-18     87128.277196
2025-04-19    100085.568231
Name: Total, dtype: float64 Lok Ma Chau Spur Line 0.06735435184718744
Date
2025-04-15    49898.161408
2025-04-16    47752.503253
2025-04-17    52608.317729
2025-04-18    78406.076395
2025-04-19    89956.543328
Name: Total, dtype: float64 Shenzhen Bay 0.0901117667880495
```

[21]:
```
step_list = [1, 1, 1, 3, 1]
result_dict = {}  #
for (name, data), step in zip(data_dict.items(), step_list):
    if name == 'Hong Kong International Airport' or name == 'Lo Wu':
        result = process_portfolio(name, data, step)
        result_dict[name] = result
        print(f"{name}    ")
        print(f'MAPE: {result["metrics"]}')
    else:
        pass
```

```
Hong Kong International Airport
MAPE:         ETS    SARIMA  AutoTBATS  DynamicOptimizedTheta      MSTL
Ensemble
1  0.040408  0.048045     0.0455               0.043696  0.040081  0.065324
Lo Wu
MAPE:        ETS    SARIMA  AutoTBATS  DynamicOptimizedTheta      MSTL  Ensemble
1  0.08068  0.069972   0.068026               0.075337  0.075878  0.094735
```

[22]:
```
total_horizon = 5  #

for name, res in result_dict.items():
    #
    forecaster = res['forecaster']
    data = res['data']
    step = res['step']
    current_step = step   #
    metrics_min = res['metrics'].loc[current_step].min()
    data_with_id = data.copy()
    data_with_id.insert(0, 'unique_id', 1)
    if forecaster.best_model_name == 'Ensemble':
        ensemble_model = res['ensemble_model']
        models_preds = []
        for model in forecaster.models:
            model_pred = rolling_predict(data_with_id, model,␣
 ↪days=total_horizon, horizon=current_step)
            print(model_pred)
            models_preds.append(model_pred)
        X_stack = np.column_stack(models_preds)
        final_pred = ensemble_model.predict(X_stack)[:total_horizon]
    else:
        best_model = forecaster.models[forecaster.model_names.index(forecaster.
 ↪best_model_name)]
        final_pred = rolling_predict(data_with_id, best_model,␣
 ↪days=total_horizon, horizon=current_step)
    print(f'{name}:  5  {final_pred} ')
    print(f'MAPE: {metrics_min} MODEL: {forecaster.best_model_name}')
```

```
Hong Kong International Airport:  5   [55670.273, 54130.62293147412,
56657.149074260364, 62817.066976302034, 60271.948274615854]
MAPE: 0.0400809193418922 MODEL: MSTL
Lo Wu:  5   [78187.88, 80333.5739353477, 80148.13705146244,
81973.45547264464, 98345.2434523564]
MAPE: 0.06802588430365338 MODEL: AutoTBATS
```