Totpal - Project 2 Code

February 15, 2026

Title: ICU Occupancy Change Prediction in California

Author: Allyson Totpal

Date: 15 February 2026

Modified By: Allyson Totpal

Description: Forecasting medium-term hospital capacity strain in California using respiratory emergency department surveillance signals.

[2]:
```
import pandas as pd
import numpy as np

# read in data
hhs = pd.read_csv('/Users/smooshii/DSC680/
↪COVID-19_Reported_Patient_Impact_and_Hospital_Capacity_by_Facility_20260208.
↪csv')
nssp = pd.read_csv('/Users/smooshii/DSC680/
↪NSSP_Emergency_Department_Visit_Trajectories_by_State_and_Sub_State_Regions-_COVID-19,_Flu,
↪csv')
cdc_dash = pd.read_csv('/Users/smooshii/DSC680/respiratory-virus-dashboard.csv')
```

```
/var/folders/_p/j05_fthn2yqfx9lkpltf1p0h0000gn/T/ipykernel_75917/1197078806.py:5
: DtypeWarning: Columns (0,3) have mixed types. Specify dtype option on import
or set low_memory=False.
  hhs = pd.read_csv('/Users/smooshii/DSC680/COVID-
19_Reported_Patient_Impact_and_Hospital_Capacity_by_Facility_20260208.csv')
```

[3]:
```
nssp.head()
```

[3]:
```
     week_end geography    county  percent_visits_combined  \
0  2022-10-01   Alabama      Bibb                      NaN
1  2022-10-01   Alabama   Calhoun                      NaN
2  2022-10-01   Alabama   Chilton                      NaN
3  2022-10-01   Alabama  Cleburne                      NaN
4  2022-10-01   Alabama     Coosa                      NaN

   percent_visits_covid  percent_visits_influenza  percent_visits_rsv  \
```

```
0                   NaN                       NaN                   NaN
1                   NaN                       NaN                   NaN
2                   NaN                       NaN                   NaN
3                   NaN                       NaN                   NaN
4                   NaN                       NaN                   NaN

  percent_visits_smoothed_combined  percent_visits_smoothed_covid  \
0                              NaN                            NaN
1                              NaN                            NaN
2                              NaN                            NaN
3                              NaN                            NaN
4                              NaN                            NaN

  percent_visits_smoothed_influenza  percent_visits_smoothed_rsv  \
0                               NaN                          NaN
1                               NaN                          NaN
2                               NaN                          NaN
3                               NaN                          NaN
4                               NaN                          NaN

   ed_trends_covid ed_trends_influenza     ed_trends_rsv  \
0  Data Unavailable    Data Unavailable  Data Unavailable
1  Data Unavailable    Data Unavailable  Data Unavailable
2  Data Unavailable    Data Unavailable  Data Unavailable
3  Data Unavailable    Data Unavailable  Data Unavailable
4  Data Unavailable    Data Unavailable  Data Unavailable

                                         hsa  \
0  Jefferson (Birmingham), AL - Shelby, AL
1    Calhoun (Anniston), AL - Cleburne, AL
2  Jefferson (Birmingham), AL - Shelby, AL
3    Calhoun (Anniston), AL - Cleburne, AL
4                 Talladega, AL - Clay, AL

                                    hsa_counties hsa_nci_id   fips  \
0  Bibb, Blount, Chilton, Cullman, Jefferson, She…        150  1,007
1                               Calhoun, Cleburne        177  1,015
2  Bibb, Blount, Chilton, Cullman, Jefferson, She…        150  1,021
3                               Calhoun, Cleburne        177  1,029
4                          Clay, Coosa, Talladega        241  1,037

  trend_source BuildNumber
0          HSA  2026-02-06
1          HSA  2026-02-06
2          HSA  2026-02-06
3          HSA  2026-02-06
4          HSA  2026-02-06
```

0.0.1 NSSP Preprocessing

[5]:
```
# standardize column names
nssp.columns = [c.strip() for c in nssp.columns]
nssp.head()

# parse dates
nssp['week_end'] = pd.to_datetime(nssp['week_end'], errors = 'coerce')

# filter to CA only
nssp = nssp[nssp['geography'].str.lower() == 'california'].copy()

# drop high granularity geography
nssp = nssp.drop(columns = ['county', 'hsa'], errors = 'ignore')
```

[6]:
```
# candidate Ed signal columns
ed_cols = ['percent_visits_influenza', 'percent_visits_rsv',␣
↪'percent_visits_combined',
           'percent_visits_smoothed_influenza', 'percent_visits_smoothed_rsv',␣
↪'percent_visits_smoothed_combined']
keep_cols = ['week_end'] + [c for c in ed_cols if c in nssp.columns]
nssp = nssp[keep_cols]

# convert to numeric
for c in ed_cols:
    if c in nssp.columns:
        nssp[c] = pd.to_numeric(nssp[c], errors = 'coerce')
```

[7]:
```
# aggregate to one row per week_end
nssp_week = (nssp.groupby('week_end', as_index = False)
             .mean()
             .sort_values('week_end')
            )
nssp_week.rename(columns = {'week_end': 'week_ending'}, inplace = True)
nssp_week.head()
```

[7]:
```
  week_ending  percent_visits_influenza  percent_visits_rsv  \
0  2022-10-01                      0.21                0.13
1  2022-10-08                      0.27                0.23
2  2022-10-15                      0.43                0.39
3  2022-10-22                      0.65                0.60
4  2022-10-29                      1.01                0.80

   percent_visits_combined  percent_visits_smoothed_influenza  \
0                     1.51                               0.15
1                     1.58                               0.21
2                     1.83                               0.31
```

```
3                      2.31                              0.45
4                      2.95                              0.70

   percent_visits_smoothed_rsv  percent_visits_smoothed_combined
0                         0.10                              1.55
1                         0.16                              1.55
2                         0.26                              1.65
3                         0.41                              1.91
4                         0.59                              2.36
```

[8]:
```
# engineer rolling means + week-over-week deltas
for c in ed_cols:
    if c in nssp_week.columns:
        nssp_week[f'{c}_roll2'] = nssp_week[c].rolling(2, min_periods = 1).
 ↪mean()
        nssp_week[f'{c}_roll3'] = nssp_week[c].rolling(3, min_periods = 1).
 ↪mean()
        nssp_week[f'{c}_wow_delta'] = nssp_week[c].diff()
```

0.0.2 CDC Dashboard Preprocessing

[10]:
```
cdc_dash['WEEKENDING'] = pd.to_datetime(cdc_dash['WEEKENDING'], errors =␣
 ↪'coerce')

# identify useful columns
cols_of_interest = ['WEEKENDING', 'FLU_ED_VISITS', 'RSV_ED_VISITS',␣
 ↪'FLU_ADMISSIONS', 'RSV_ADMISSIONS', 'POP']

cdc_dash = cdc_dash[[c for c in cols_of_interest if c in cdc_dash.columns]]
```

[11]:
```
# convert to numeric
for c in cdc_dash.columns:
    if c != 'WEEKENDING':
        cdc_dash[c] = pd.to_numeric(cdc_dash[c], errors = 'coerce')
```

[12]:
```
# aggregate to state-week
cdc_dash_week = (cdc_dash.groupby('WEEKENDING', as_index = False)
            .sum(numeric_only = True)
            .sort_values('WEEKENDING')
           )
cdc_dash_week.rename(columns = {'WEEKENDING': 'week_ending'}, inplace = True)
cdc_dash_week.head()
```

[12]:
```
  week_ending  FLU_ED_VISITS  RSV_ED_VISITS  POP
0  2023-07-08            0.1            0.0  0.0
1  2023-07-15            0.1            0.0  0.0
2  2023-07-22            0.1            0.0  0.0
```

```
3  2023-07-29           0.1            0.0  0.0
4  2023-08-05           0.1            0.0  0.0
```

[13]:
```
# compute per-100k rates if POP exists
if 'POP' in cdc_dash_week.columns:
    pop = cdc_dash_week['POP'].replace(0, np.nan)

    for c in ['FLU_ED_VISITS', 'RSV_ED_VISITS', 'FLU_ADMISSIONS',␣
 ↪'RSV_ADMISSIONS']:
        if c in cdc_dash_week.columns:
            cdc_dash_week[f'{c}_per_100k'] = (cdc_dash_week[c] / pop) * 100_000
```

[14]:
```
# rolling features on rate columns
rate_cols = [c for c in cdc_dash_week.columns if c.endswith('_per_100k')]

for c in rate_cols:
    cdc_dash_week[f'{c}_roll2'] = cdc_dash_week[c].rolling(2, min_periods = 1).
 ↪mean()
    cdc_dash_week[f'{c}_roll3'] = cdc_dash_week[c].rolling(3, min_periods = 1).
 ↪mean()
    cdc_dash_week[f'{c}_wow_delta'] = cdc_dash_week[c].diff()
```

0.0.3 HHS Preprocessing

[16]:
```
# select columns needed
hhs_cols = ['collection_week', 'state',␣
 ↪'all_adult_hospital_inpatient_beds_7_day_sum',
            'inpatient_beds_used_7_day_sum',␣
 ↪'staffed_adult_icu_bed_occupancy_7_day_avg']
hhs = hhs[hhs_cols].copy()
```

[17]:
```
# parse week & filter to CA
hhs.loc[:, 'collection_week'] = pd.to_datetime(hhs.loc[:, 'collection_week'],␣
 ↪errors = 'coerce')
hhs = hhs[hhs['state'] == 'CA'].copy()

# columns to clean
count_cols = ['all_adult_hospital_inpatient_beds_7_day_sum',␣
 ↪'inpatient_beds_used_7_day_sum']
raw_rate_cols = 'staffed_adult_icu_bed_occupancy_7_day_avg'
```

[18]:
```
def _clean_numeric_series(s: pd.Series) -> pd.Series:
    '''robust numeric cleaning'''
    return pd.to_numeric(
        s.astype(str)
        .str.replace(',', '', regex = False)
        .str.replace('%', '', regex = False)
```

```
        .str.strip()
        .replace({'': np.nan, 'nan': np.nan, 'None': np.nan}),
        errors = 'coerce'
    )
```

[19]:
```
# clean numeric fields
for c in count_cols + [raw_rate_cols]:
    hhs[c] = _clean_numeric_series(hhs[c])

# aggregate to week
hhs_week = (hhs.groupby('collection_week', as_index = False)
            .agg({**{c: 'sum' for c in count_cols},
                  raw_rate_cols: 'mean'})
            .sort_values('collection_week')
            .rename(columns = {'collection_week': 'week_ending'})
           )
```

[20]:
```
hhs_week['icu_occ_raw'] = hhs_week[raw_rate_cols].abs() / 1_000_000

# winsorize ICU occupancy at p99
cap_99 = hhs_week['icu_occ_raw'].quantile(0.99)
hhs_week['icu_occ_wins'] = hhs_week['icu_occ_raw'].clip(upper = cap_99)

# final ICU occupancy feature
hhs_week['icu_occ'] = hhs_week['icu_occ_wins']

# inpatient occupancy ratio
hhs_week['inpatient_occ'] = (hhs_week['inpatient_beds_used_7_day_sum'] /
                                         ␣
 ↪hhs_week['all_adult_hospital_inpatient_beds_7_day_sum'].replace(0, np.nan))

# high strain flag
p95 = hhs_week['icu_occ_wins'].quantile(0.95)
hhs_week['high_strain'] = (hhs_week['icu_occ_wins'] >= p95).astype(int)

# create standardized Monday-start week key for joining
hhs_week['week'] = hhs_week['week_ending'].dt.to_period('W-MON').dt.start_time
```

[21]:
```
print('Weeks:', len(hhs_week))
print('Non-missing icu_occ:', int(hhs_week['icu_occ'].notna().sum()))
print('ICU occ wins cap (p99):', cap_99)
print(hhs_week['icu_occ'].describe())
print('p95 threshold:', p95)
print(hhs_week['high_strain'].value_counts())

hhs_week.head()
```

```
Weeks: 218
```

```
Non-missing icu_occ: 198
ICU occ wins cap (p99): 0.1930038756231166
count    198.000000
mean       0.140260
std        0.029996
min        0.059637
25%        0.121798
50%        0.142028
75%        0.163716
max        0.193004
Name: icu_occ, dtype: float64
p95 threshold: 0.18143368775902988
high_strain
0    208
1     10
Name: count, dtype: int64
```

[21]:

```
   week_ending  all_adult_hospital_inpatient_beds_7_day_sum  \
0   2020-02-02                                          0.0
1   2020-03-01                                          0.0
2   2020-03-08                                          0.0
3   2020-03-15                                          0.0
4   2020-03-22                                          0.0

   inpatient_beds_used_7_day_sum  staffed_adult_icu_bed_occupancy_7_day_avg  \
0                           77.0                                        NaN
1                           50.0                                        NaN
2                          594.0                                        NaN
3                         2127.0                                        NaN
4                         5973.0                                        NaN

   icu_occ_raw  icu_occ_wins  icu_occ  inpatient_occ  high_strain       week
0          NaN           NaN      NaN            NaN            0 2020-01-28
1          NaN           NaN      NaN            NaN            0 2020-02-25
2          NaN           NaN      NaN            NaN            0 2020-03-03
3          NaN           NaN      NaN            NaN            0 2020-03-10
4          NaN           NaN      NaN            NaN            0 2020-03-17
```

0.0.4 Merge Weekly Datasets

[23]:

```
# merge on week_ending
hhs_week['week']  = hhs_week['week_ending'].dt.to_period('W-MON').dt.start_time
nssp_week['week'] = nssp_week['week_ending'].dt.to_period('W-MON').dt.start_time
cdc_dash_week['week']  = cdc_dash_week['week_ending'].dt.to_period('W-MON').dt.
  ↪start_time
```

```
[24]: data = (hhs_week
             .merge(nssp_week, on = 'week', how = 'left')
             .sort_values('week')
             .reset_index(drop = True)
            )
```

0.0.5 Build Forecasting and Modeling

```
[26]: # create forecast target (3 weeks ahead)
forecast_horizon = 3
data['icu_occ_t_plus_3w'] = data['icu_occ'].shift(-forecast_horizon)

# drop rows where future target is missing
model_df = data.dropna(subset = ['icu_occ_t_plus_3w']).copy()
print('Model Rows: ', len(model_df))
```

```
Model Rows:  198
```

```
[27]: # select predictors
feature_candidates = ['icu_occ', 'FLU_ED_VISITS_per_100k',␣
 ↪'RSV_ED_VISITS_per_100k',
                      'FLU_ED_VISITS_per_100k_roll2',␣
 ↪'FLU_ED_VISITS_per_100k_roll3',
                      'RSV_ED_VISITS_per_100k_roll2',␣
 ↪'RSV_ED_VISITS_per_100k_roll3']
features = [f for f in feature_candidates if f in model_df.columns]
print('Using features: ', features)
```

```
Using features:  ['icu_occ']
```

```
[28]: # build x and y
x = model_df[features].copy()
y = model_df['icu_occ_t_plus_3w']

# drop rows with missing predictors
mask = x.notna().all(axis = 1)
x = x[mask]
y = y[mask]

print('Final Modeling Rows: ', len(x))
```

```
Final Modeling Rows:  195
```

```
[29]: # train/test split
split_idx = int(len(x) * 0.8)

x_train = x.iloc[:split_idx]
x_test = x.iloc[split_idx:]
```

```
y_train = y.iloc[:split_idx]
y_test = y.iloc[split_idx:]
```

[30]:
```
from sklearn.linear_model import Ridge
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.metrics import mean_squared_error, mean_absolute_error

# ridge regression (standardized)
ridge_model = Pipeline([('scaler', StandardScaler()),
                        ('ridge', Ridge(alpha = 1.0))])
ridge_model.fit(x_train, y_train)
ridge_preds = ridge_model.predict(x_test)

print('Ridge RMSE: ', mean_squared_error(y_test, ridge_preds, squared = False))
print('Ridge MAE: ', mean_absolute_error(y_test, ridge_preds))
```

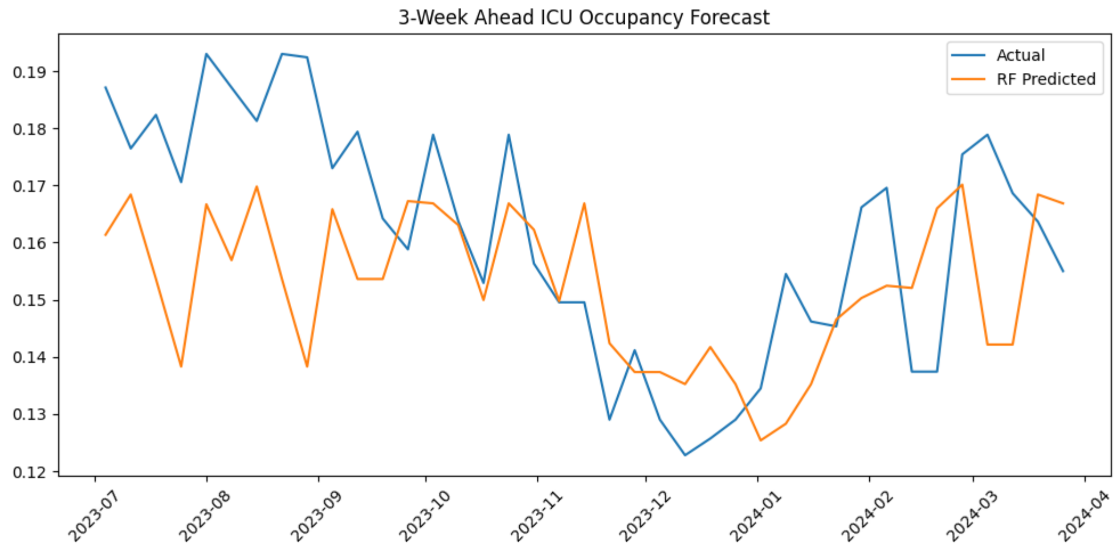
```
Ridge RMSE:  0.01635486835867053
Ridge MAE:  0.013185163422887971
```

[31]:
```
from sklearn.ensemble import RandomForestRegressor

# random forest regression
rf_model = RandomForestRegressor(n_estimators = 300,
                                 max_depth = 6,
                                 random_state = 42)
rf_model.fit(x_train, y_train)
rf_preds = rf_model.predict(x_test)
print('RF RMSE: ', mean_squared_error(y_test, rf_preds, squared = False))
print('RF MAE: ', mean_absolute_error(y_test, rf_preds))
```

```
RF RMSE:  0.02010146066469092
RF MAE:  0.01610909346292938
```

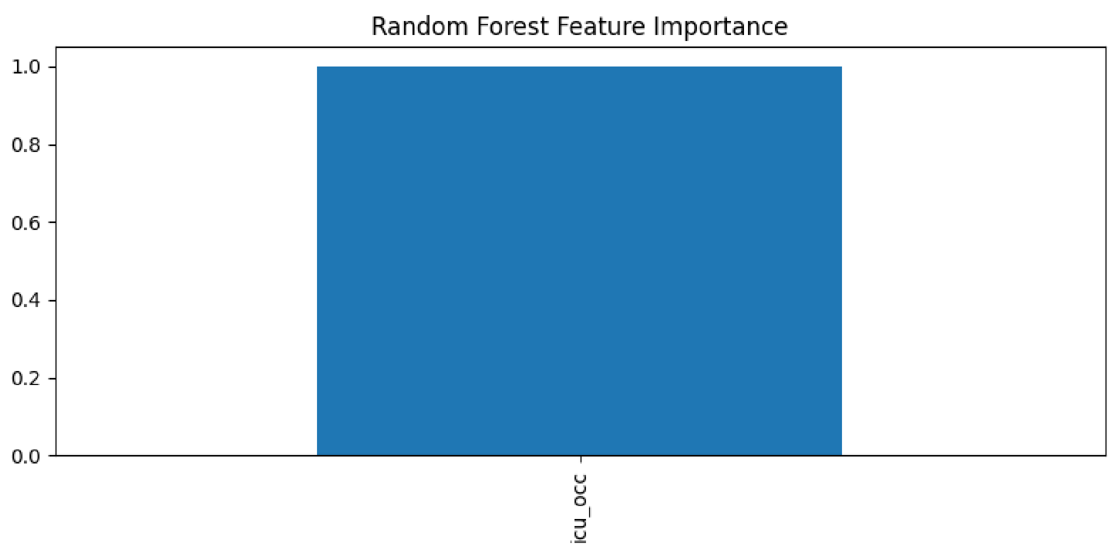
[32]:
```
import matplotlib.pyplot as plt

# make sure y_test and predictions are aligned
test_weeks = model_df.loc[y_test.index, 'week']

# plot actual vs predicted change
plt.figure(figsize = (10, 5))
plt.plot(test_weeks, y_test.values, label = 'Actual')
plt.plot(test_weeks, rf_preds, label = 'RF Predicted')
plt.legend()
plt.title('3-Week Ahead ICU Occupancy Forecast')
plt.xticks(rotation = 45)
plt.tight_layout()
```

```
plt.show()
```



[33]:

```python
import pandas as pd

importances = pd.Series(rf_model.feature_importances_,
                        index = x.columns).sort_values(ascending = False)

importances.plot(kind = 'bar', figsize = (8,4))
plt.title('Random Forest Feature Importance')
plt.tight_layout()
plt.show()
```



[34]:
```
# update forecast to 7-weeks
forecast_horizon = 7

data['icu_occ_t_plus_7w'] = data['icu_occ'].shift(-7)
model_df = data.dropna(subset = features + ['icu_occ_t_plus_7w']).copy()
```

[35]:
```
# select predictors
feature_candidates = ['icu_occ', 'FLU_ED_VISITS_per_100k',␣
 ↪'RSV_ED_VISITS_per_100k',
                      'FLU_ED_VISITS_per_100k_roll2',␣
 ↪'FLU_ED_VISITS_per_100k_roll3',
                      'RSV_ED_VISITS_per_100k_roll2',␣
 ↪'RSV_ED_VISITS_per_100k_roll3']
features = [f for f in feature_candidates if f in model_df.columns]
print('Using features: ', features)
```

```
Using features:  ['icu_occ']
```

[36]:
```
x = model_df[features].copy()
y = model_df['icu_occ_t_plus_3w']

# drop rows with missing predictors
mask = x.notna().all(axis = 1)
x = x[mask]
y = y[mask]

print('Final Modeling Rows: ', len(x))

# train/test split
split_idx = int(len(x) * 0.8)

x_train = x.iloc[:split_idx]
x_test = x.iloc[split_idx:]

y_train = y.iloc[:split_idx]
y_test = y.iloc[split_idx:]
```

```
Final Modeling Rows:  191
```

[37]:
```
# build ridge model
ridge_model = Pipeline([('scaler', StandardScaler()),
                        ('ridge', Ridge(alpha = 1.0))])
ridge_model.fit(x_train, y_train)
ridge_preds = ridge_model.predict(x_test)

print('Ridge RMSE: ', mean_squared_error(y_test, ridge_preds, squared = False))
```

```
print('Ridge MAE: ', mean_absolute_error(y_test, ridge_preds))
```

```
Ridge RMSE:  0.015486797449980468
Ridge MAE:  0.012732300156640845
```

[38]:
```
# build random forest model
rf_model = RandomForestRegressor(n_estimators = 300,
                                 max_depth = 6,
                                 random_state = 42)
rf_model.fit(x_train, y_train)
rf_preds = rf_model.predict(x_test)
print('RF RMSE: ', mean_squared_error(y_test, rf_preds, squared = False))
print('RF MAE: ', mean_absolute_error(y_test, rf_preds))
```

```
RF RMSE:  0.020534437222615023
RF MAE:  0.01641447692032386
```

[39]:
```
data = data.sort_values('week').reset_index(drop=True)

# keep only rows where current ICU occupancy exists
data2 = data.dropna(subset = ['icu_occ']).copy()

FORECAST_HORIZON = 7
data2['icu_occ_t_plus_7w'] = data2['icu_occ'].shift(-FORECAST_HORIZON)
data2['icu_change_7w'] = data2['icu_occ_t_plus_7w'] - data2['icu_occ']

print(data2[['week', 'icu_occ', 'icu_occ_t_plus_7w', 'icu_change_7w']].head(12))
print('Rows with target available:', data2['icu_change_7w'].notna().sum())
```

```
          week   icu_occ  icu_occ_t_plus_7w  icu_change_7w
20 2020-07-07  0.084892           0.154956       0.070065
21 2020-07-14  0.105866           0.119519       0.013653
22 2020-07-21  0.137915           0.148674       0.010759
23 2020-07-28  0.134487           0.142428       0.007940
24 2020-08-04  0.133706           0.143664       0.009959
25 2020-08-11  0.128265           0.150275       0.022011
26 2020-08-18  0.136613           0.158488       0.021875
27 2020-08-25  0.154956           0.142436      -0.012520
28 2020-09-01  0.119519           0.156236       0.036717
29 2020-09-08  0.148674           0.150128       0.001454
30 2020-09-15  0.142428           0.130297      -0.012131
31 2020-09-22  0.143664           0.121798      -0.021866
Rows with target available: 191
```

[40]:
```
# sort chronologically
data = data.sort_values('week').reset_index(drop=True)

# keep only rows where ICU occupancy exists
data2 = data.dropna(subset=['icu_occ']).copy()
```

```python
print('Rows with valid ICU:', len(data2))

# create 7-week horizon change
FORECAST_HORIZON = 7

data2['icu_occ_t_plus_7w'] = data2['icu_occ'].shift(-FORECAST_HORIZON)
data2['icu_change_7w'] = data2['icu_occ_t_plus_7w'] - data2['icu_occ']

# drop rows without future target
model_df = data2.dropna(subset = ['icu_change_7w']).copy()

print('Model rows:', len(model_df))
print('Date range:',
      model_df['week'].min(),
      '→',
      model_df['week'].max())
```

```
Rows with valid ICU: 198
Model rows: 191
Date range: 2020-07-07 00:00:00 → 2024-02-27 00:00:00
```

[41]:
```python
# resplit
features = ['icu_occ', 'FLU_ED_VISITS_per_100k', 'RSV_ED_VISITS_per_100k']

features = [f for f in features if f in model_df.columns]

split_idx = int(len(model_df) * 0.8)

train_df = model_df.iloc[:split_idx]
test_df  = model_df.iloc[split_idx:]

X_train = train_df[features]
y_train = train_df['icu_change_7w']

X_test  = test_df[features]
y_test  = test_df['icu_change_7w']
```

[42]:
```python
from sklearn.metrics import mean_squared_error, mean_absolute_error

# naive baseline: predict zero change
naive_preds = np.zeros(len(y_test))

naive_rmse = mean_squared_error(y_test, naive_preds, squared=False)
naive_mae  = mean_absolute_error(y_test, naive_preds)

print('Naive RMSE:', naive_rmse)
print('Naive MAE:', naive_mae)
```

```
Naive RMSE: 0.024170987362290043
Naive MAE: 0.02001658917467064
```

[43]:

```python
# build new ridge model
ridge_model = Pipeline([('scaler', StandardScaler()),
                        ('ridge', Ridge(alpha = 1.0))])

ridge_model.fit(X_train, y_train)
ridge_preds = ridge_model.predict(X_test)

ridge_rmse = mean_squared_error(y_test, ridge_preds, squared = False)
ridge_mae  = mean_absolute_error(y_test, ridge_preds)

print('Ridge RMSE:', ridge_rmse)
print('Ridge MAE:', ridge_mae)
```

```
Ridge RMSE: 0.02490003351377585
Ridge MAE: 0.020886145624497515
```

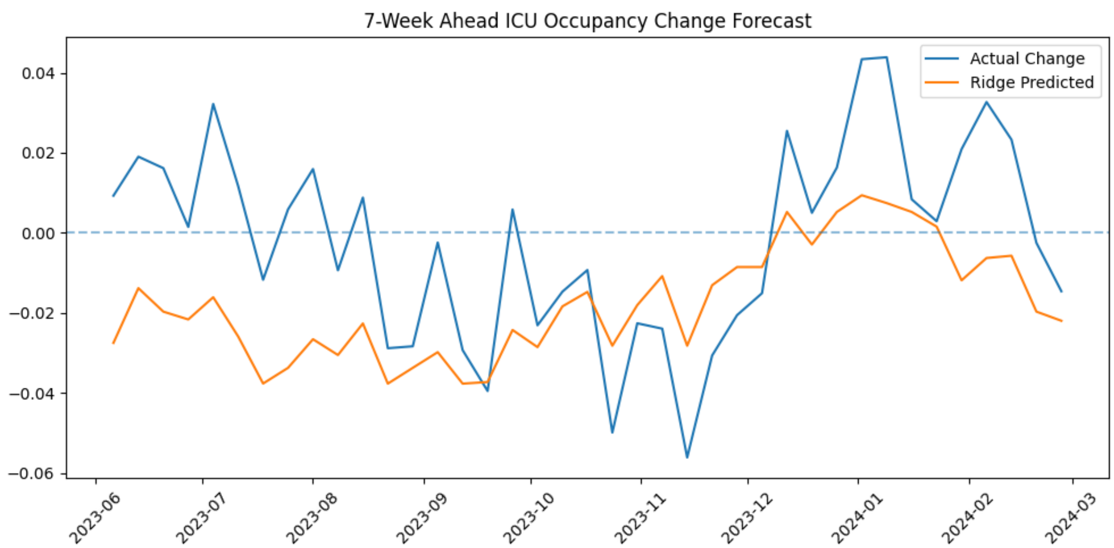
[44]:

```python
from sklearn.ensemble import RandomForestRegressor

# build new random forest model
rf_model = RandomForestRegressor(n_estimators = 300,
                                 max_depth = 6,
                                 random_state = 42)

rf_model.fit(X_train, y_train)
rf_preds = rf_model.predict(X_test)

rf_rmse = mean_squared_error(y_test, rf_preds, squared=False)
rf_mae  = mean_absolute_error(y_test, rf_preds)

print('RF RMSE:', rf_rmse)
print('RF MAE:', rf_mae)
```

```
RF RMSE: 0.02845821680279954
RF MAE: 0.021704250602392942
```

[45]:

```python
# plot actual vs predicted in 7-week ahead model
plt.figure(figsize = (10, 5))
plt.plot(test_df['week'], y_test.values, label = 'Actual Change')
plt.plot(test_df['week'], ridge_preds, label = 'Ridge Predicted')
plt.axhline(0, linestyle = '--', alpha = 0.5)
plt.legend()
plt.title('7-Week Ahead ICU Occupancy Change Forecast')
plt.xticks(rotation = 45)
plt.tight_layout()
plt.show()
```



[46]:
```python
# identify ED-rate columns that actually exist
flu_col = next((c for c in model_df.columns if 'flu' in c.lower() and␣
↪'per_100k' in c.lower()), None)
rsv_col = next((c for c in model_df.columns if 'rsv' in c.lower() and␣
↪'per_100k' in c.lower()), None)

print('Detected flu column:', flu_col)
print('Detected rsv column:', rsv_col)

# add lagged ICU levels
model_df = model_df.sort_values('week').reset_index(drop = True)
model_df['icu_lag1'] = model_df['icu_occ'].shift(1)
model_df['icu_lag2'] = model_df['icu_occ'].shift(2)

# add lagged ED signals if columns exist
if flu_col is not None:
    model_df['flu_lag1'] = model_df[flu_col].shift(1)
    model_df['flu_lag2'] = model_df[flu_col].shift(2)

if rsv_col is not None:
    model_df['rsv_lag1'] = model_df[rsv_col].shift(1)
    model_df['rsv_lag2'] = model_df[rsv_col].shift(2)

# drop rows with new NaNs from lags/targets
model_df = model_df.dropna(subset = ['icu_change_7w', 'icu_occ', 'icu_lag1',␣
↪'icu_lag2']).copy()
if flu_col is not None:
    model_df = model_df.dropna(subset = ['flu_lag1', 'flu_lag2']).copy()
```

```
if rsv_col is not None:
    model_df = model_df.dropna(subset = ['rsv_lag1', 'rsv_lag2']).copy()

print('Rows after adding lags:', len(model_df))
```

```
Detected flu column: None
Detected rsv column: None
Rows after adding lags: 189
```

[47]:
```
features = ['icu_occ', 'icu_lag1', 'icu_lag2']

if flu_col is not None:
    features += [flu_col, 'flu_lag1', 'flu_lag2']
if rsv_col is not None:
    features += [rsv_col, 'rsv_lag1', 'rsv_lag2']

print('Features used:', features)
```

```
Features used: ['icu_occ', 'icu_lag1', 'icu_lag2']
```

[48]:
```
flu_col = 'percent_visits_smoothed_influenza'
rsv_col = 'percent_visits_smoothed_rsv'

print('Using flu column:', flu_col)
print('Using rsv column:', rsv_col)
```

```
Using flu column: percent_visits_smoothed_influenza
Using rsv column: percent_visits_smoothed_rsv
```

[49]:
```
model_df = model_df.sort_values('week').reset_index(drop=True)

# ICU lags
model_df['icu_lag1'] = model_df['icu_occ'].shift(1)
model_df['icu_lag2'] = model_df['icu_occ'].shift(2)

# ED lags
model_df['flu_lag1'] = model_df[flu_col].shift(1)
model_df['flu_lag2'] = model_df[flu_col].shift(2)

model_df['rsv_lag1'] = model_df[rsv_col].shift(1)
model_df['rsv_lag2'] = model_df[rsv_col].shift(2)

# drop rows with lag-induced NaNs
model_df = model_df.dropna().copy()

print('Rows after adding lags:', len(model_df))
```

```
Rows after adding lags: 69
```

[50]:
```
features = ['icu_occ', 'icu_lag1', 'icu_lag2', flu_col,
            'flu_lag1', 'flu_lag2', rsv_col, 'rsv_lag1', 'rsv_lag2']

print('Feature count:', len(features))
```

```
Feature count: 9
```

[51]:
```
split_idx = int(len(model_df) * 0.8)

train_df = model_df.iloc[:split_idx]
test_df  = model_df.iloc[split_idx:]

X_train = train_df[features]
y_train = train_df['icu_change_7w']

X_test  = test_df[features]
y_test  = test_df['icu_change_7w']
```

[52]:
```
naive_preds = np.zeros(len(y_test))

print('Naive RMSE:', mean_squared_error(y_test, naive_preds, squared=False))
print('Naive MAE:', mean_absolute_error(y_test, naive_preds))
```

```
Naive RMSE: 0.023513781015979705
Naive MAE: 0.019637858420002756
```

[53]:
```
ridge_model = Pipeline([('scaler', StandardScaler()),
                        ('ridge', Ridge(alpha = 1.0))])

ridge_model.fit(X_train, y_train)
ridge_preds = ridge_model.predict(X_test)

print('Ridge RMSE:', mean_squared_error(y_test, ridge_preds, squared = False))
print('Ridge MAE:', mean_absolute_error(y_test, ridge_preds))
```

```
Ridge RMSE: 0.02478472755135359
Ridge MAE: 0.020009696341092075
```

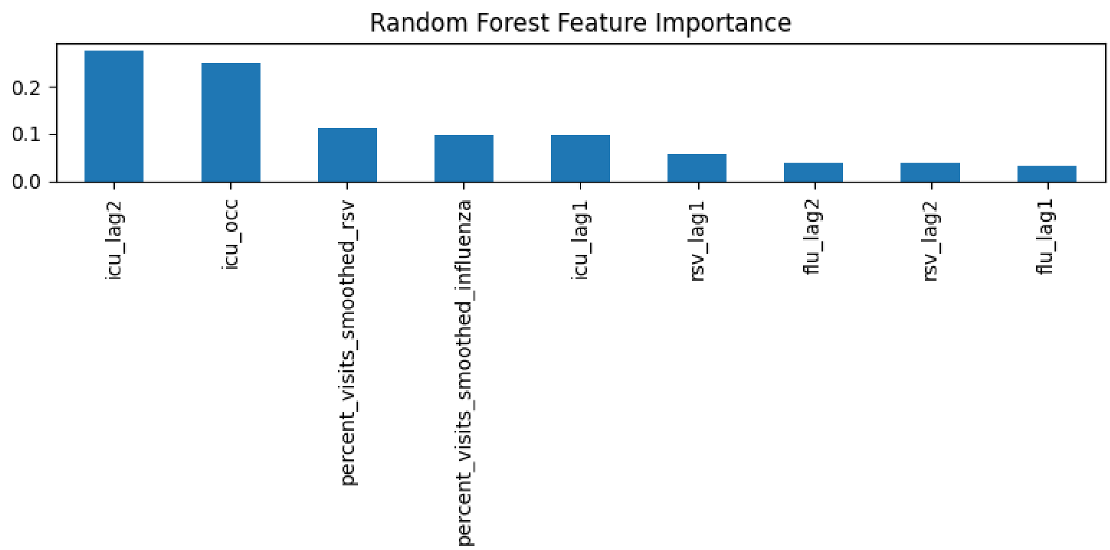
[54]:
```
rf_model = RandomForestRegressor(n_estimators = 300,
                                 max_depth = 6,
                                 random_state = 42)

rf_model.fit(X_train, y_train)
rf_preds = rf_model.predict(X_test)

print('RF RMSE:', mean_squared_error(y_test, rf_preds, squared = False))
print('RF MAE:', mean_absolute_error(y_test, rf_preds))
```

```
RF RMSE: 0.021597451258293918
```

```
RF MAE: 0.017485906630914816
```

[55]:
```python
importances = pd.Series(rf_model.feature_importances_,
                        index = features).sort_values(ascending = False)

print(importances)

plt.figure(figsize = (8, 4))
importances.plot(kind = 'bar')
plt.title('Random Forest Feature Importance')
plt.tight_layout()
plt.show()
```

```
icu_lag2                             0.278436
icu_occ                              0.249924
percent_visits_smoothed_rsv          0.110652
percent_visits_smoothed_influenza    0.097884
icu_lag1                             0.097785
rsv_lag1                             0.056424
flu_lag2                             0.038021
rsv_lag2                             0.037510
flu_lag1                             0.033363
dtype: float64
```



[56]:
```python
# Create directional target
model_df['increase_7w'] = (model_df['icu_change_7w'] > 0).astype(int)
print(model_df['increase_7w'].value_counts())
```

```
increase_7w
1    42
```

```
0    27
Name: count, dtype: int64
```

[57]:
```
features = ['icu_occ', 'icu_lag1', 'icu_lag2',␣
↪'percent_visits_smoothed_influenza',
            'flu_lag1', 'flu_lag2', 'percent_visits_smoothed_rsv', 'rsv_lag1',␣
↪'rsv_lag2']

model_df = model_df.sort_values('week').reset_index(drop = True)

split_idx = int(len(model_df) * 0.8)

train_df = model_df.iloc[:split_idx]
test_df  = model_df.iloc[split_idx:]

X_train = train_df[features]
y_train = train_df['increase_7w']

X_test  = test_df[features]
y_test  = test_df['increase_7w']
```

[58]:
```
from sklearn.metrics import accuracy_score

majority_class = y_train.mode()[0]
baseline_preds = np.full(len(y_test), majority_class)

print('Baseline Accuracy:', accuracy_score(y_test, baseline_preds))
```

```
Baseline Accuracy: 0.7142857142857143
```

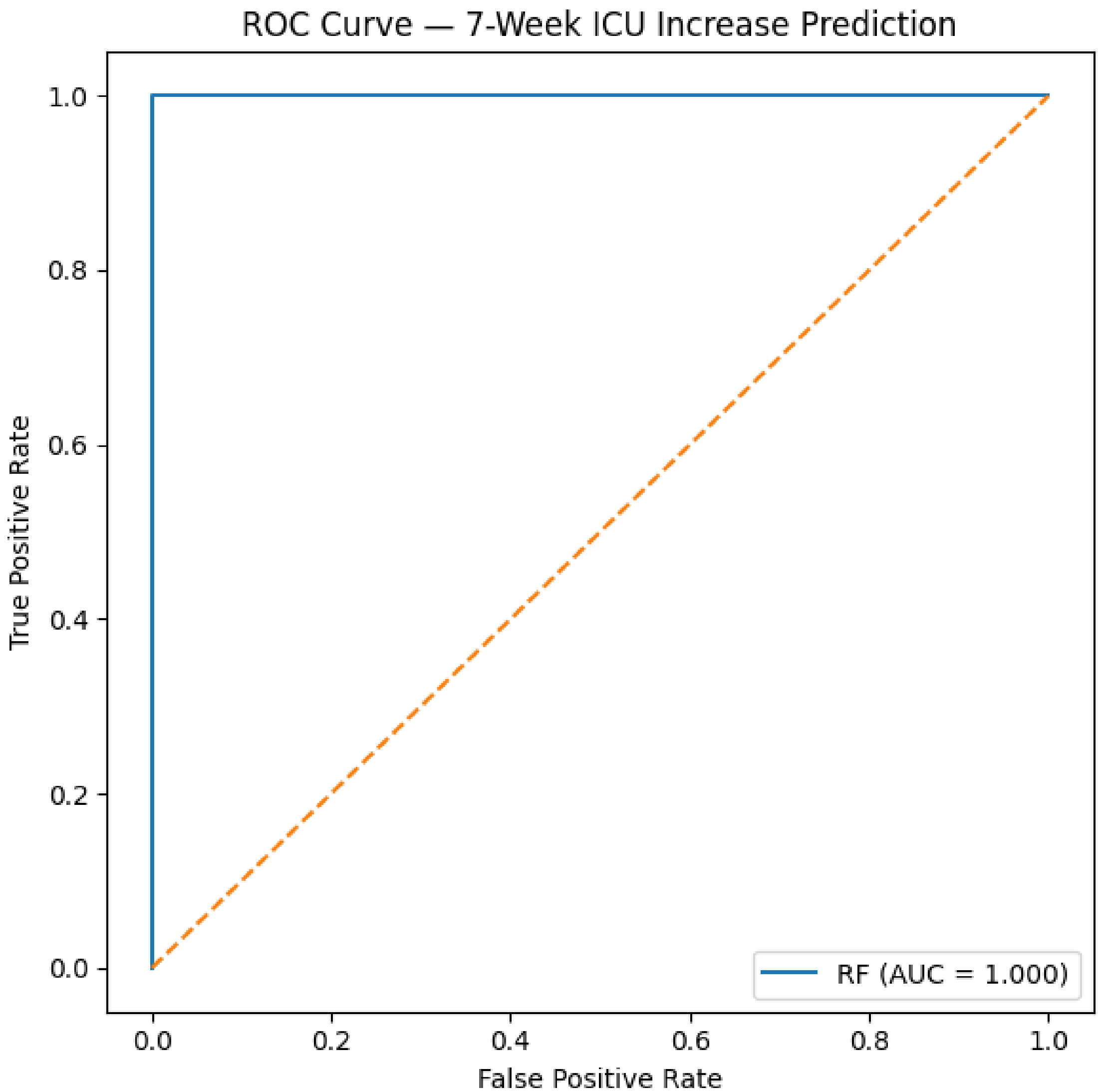
[59]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import (accuracy_score, roc_auc_score,␣
↪classification_report, confusion_matrix)

rf_clf = RandomForestClassifier(n_estimators = 400,
                                max_depth = 6,
                                random_state = 42)

rf_clf.fit(X_train, y_train)

rf_probs = rf_clf.predict_proba(X_test)[:, 1]
rf_preds = (rf_probs > 0.5).astype(int)

print('RF Accuracy:', accuracy_score(y_test, rf_preds))
print('RF ROC-AUC:', roc_auc_score(y_test, rf_probs))

print('\nConfusion Matrix:')
print(confusion_matrix(y_test, rf_preds))
```

```python
print('\nClassification Report:')
print(classification_report(y_test, rf_preds))
```

```
RF Accuracy: 0.8571428571428571
RF ROC-AUC: 1.0

Confusion Matrix:
[[4 0]
 [2 8]]

Classification Report:
              precision    recall  f1-score   support

           0       0.67      1.00      0.80         4
           1       1.00      0.80      0.89        10

    accuracy                           0.86        14
   macro avg       0.83      0.90      0.84        14
weighted avg       0.90      0.86      0.86        14
```

[60]:

```python
from sklearn.metrics import roc_curve

fpr, tpr, _ = roc_curve(y_test, rf_probs)

plt.figure(figsize = (6, 6))
plt.plot(fpr, tpr, label = f'RF (AUC = {roc_auc_score(y_test, rf_probs):.3f})')
plt.plot([0,1], [0,1], linestyle = '--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve - 7-Week ICU Increase Prediction')
plt.legend()
plt.tight_layout()
plt.show()
```

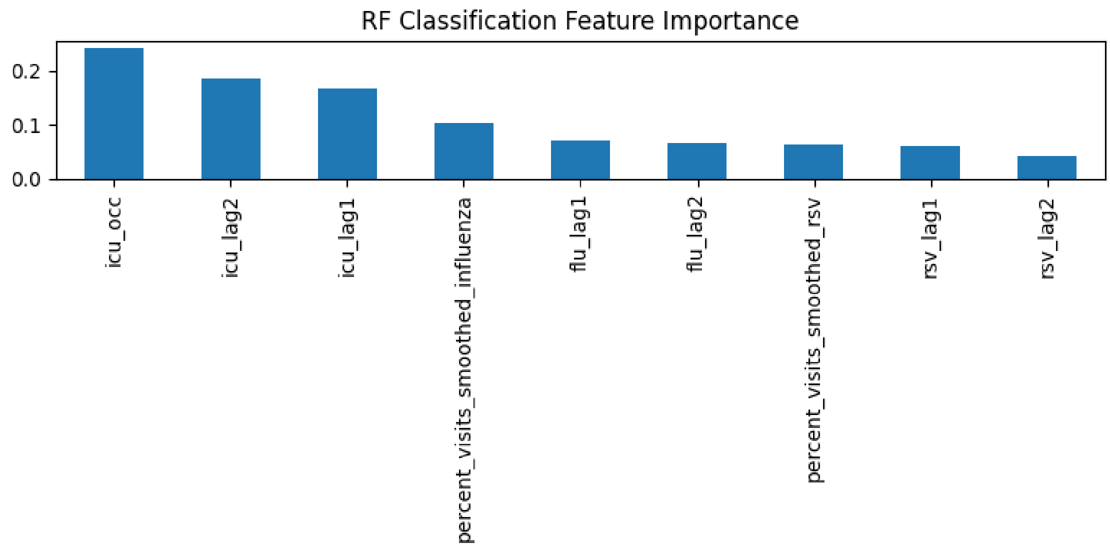


[61]:

```
clf_importance = pd.Series(rf_clf.feature_importances_, index=features).
 ↪sort_values(ascending = False)

print(clf_importance)

clf_importance.plot(kind = 'bar', figsize = (8, 4))
plt.title('RF Classification Feature Importance')
plt.tight_layout()
plt.show()
```

```
icu_occ                          0.243025
icu_lag2                         0.185907
icu_lag1                         0.167263
```

```
percent_visits_smoothed_influenza    0.102939
flu_lag1                             0.070255
flu_lag2                             0.066988
percent_visits_smoothed_rsv          0.064068
rsv_lag1                             0.059048
rsv_lag2                             0.040506
dtype: float64
```



[ ]: