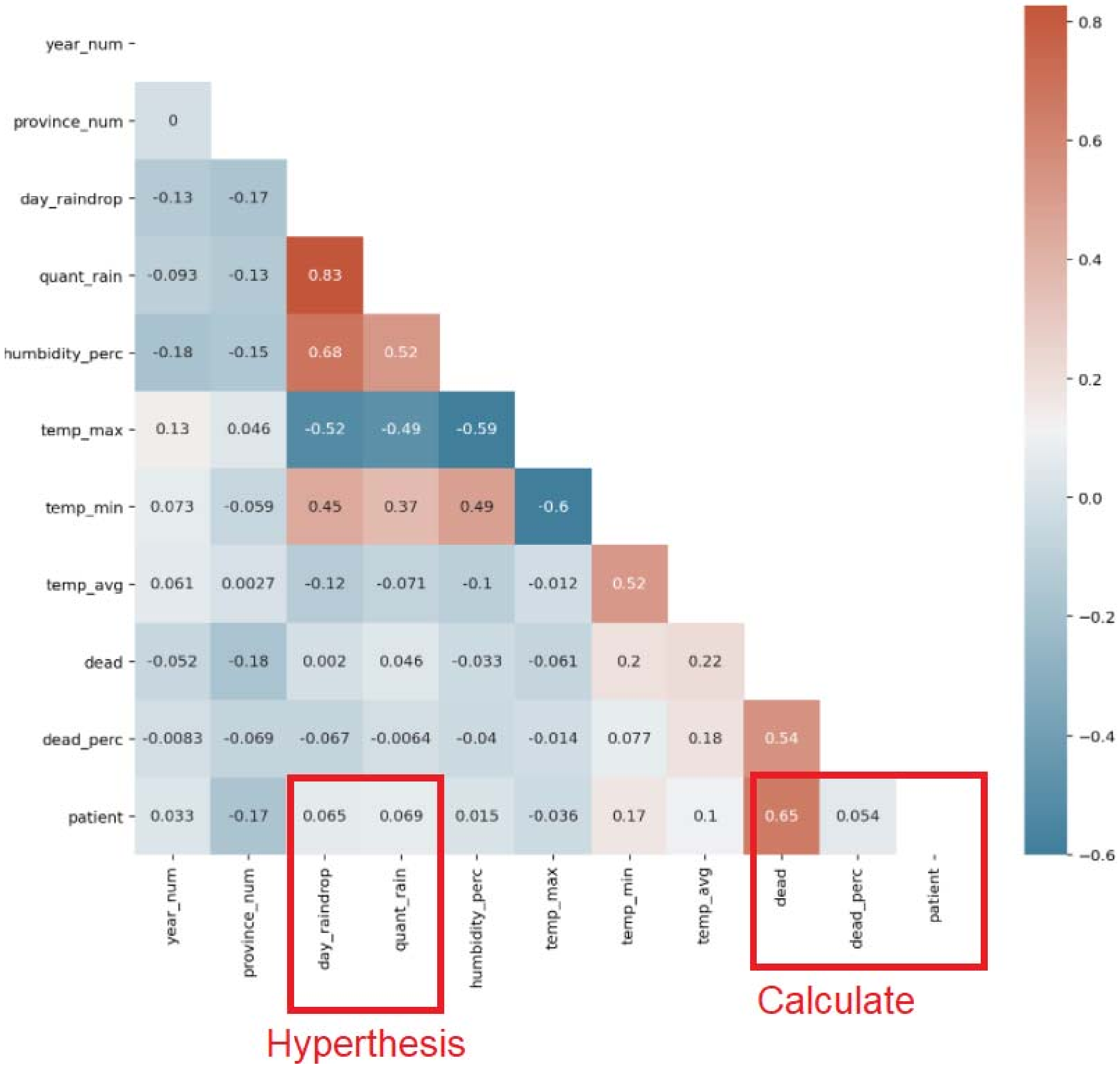
Regression model analysis report

Dataset: dengue_preprocessing.csv

Features:

1. year_num: code of year 2016 – 2020 (1-5)
2. province_num: code of province 1- 77
3. day_raindrop: how many days of rain drop in 365 day
4. quant_rain: total quantity rainwater in unit of millimeters
5. humbidity_perc: average % humbidity in each province, each year
6. temp_max: maximum temp in each province, each year
7. temp_min: minimum temp in each province, each year
8. temp_avg: average temp in each province, each year
9. dead: amount of dead person from dengue
10. dead_perc: percent of dead person from dengue
11. patient: amount of patient from dengue (target variable)

What we want to know? **Rain has affect to dengue patient or not?**



Model testing

1. Linear Regression (original)
2. Features selection: Recursive Feature Elimination (RFE) การกำจัดย้อนกลับ - เริ่มต้นด้วยตัวทำนายทั้งหมดและกำจัดทีละตัวซ้ำ ๆ หนึ่งในอัลกอริทึมที่ได้รับความนิยมมากที่สุดคือ Recursive Feature Elimination (RFE) ซึ่งกำจัดตัวทำนายที่มีความสำคัญน้อยกว่าโดยพิจารณาจากการจัดอันดับความสำคัญของคุณลักษณะ
3. GridSearchCV การค้นหาแบบกริดเป็นเทคนิคในการปรับแต่งไฮเปอร์พารามิเตอร์ที่อาจช่วยในการสร้างโมเดลและประเมินโมเดลสำหรับการรวมพารามิเตอร์อัลกอริทึมต่อตาราง
4. K-folds Cross Validation(+Repeats)
5. Polynomial regression

1.Linear Regression (original)

Features selection for Regression Model

In [22]:
```
df_pre.columns
```

Out[22]:
```
Index(['year_num', 'province_num', 'day_raindrop', 'quant_rain',
       'humbidity_perc', 'temp_max', 'temp_min', 'temp_avg', 'dead',
       'dead_perc', 'patient'],
      dtype='object')
```

In [23]:
```
X = df_pre[['year_num', 'province_num', 'day_raindrop', 'quant_rain', 'humbidity_perc','temp_max', 'temp_min', 'temp_avg', 'dead
        ]]
y = df_pre['patient']

#
```

Train Test Split

This step we will separate data to train (training set) and การ test (testing set)

- training set use for train model
- testing set use for test model or call that Evaluation

In [24]:
```
from sklearn.model_selection import train_test_split
```

In [25]:
```
# Train dataset 80% and Test dataset 20%.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=17)
```

```
from sklearn import metrics
from sklearn.metrics import mean_squared_error

print('R squared of Training Set: {:.2f}'.format(lm.score(X_train,y_train)*100))
print('R squared of Testing Set: {:.2f}'.format(lm.score(X_test,y_test)*100))
print('Mean Absolute Error (MAE): {:.4f}'.format(metrics.mean_absolute_error(y_test, predictions)))
print('Root Mean Squared Error (RMSE): {:.4f}'.format(np.sqrt(metrics.mean_squared_error(y_test, predictions))))
```

```
R squared of Training Set: 52.46
R squared of Testing Set: 64.26
Mean Absolute Error (MAE): 621.1161
Root Mean Squared Error (RMSE): 960.8271
```

In [35]:
```
#Actual value and the predicted value
mlr_diff = pd.DataFrame({'Actual value': y_test, 'Predicted value': predictions})
mlr_diff.head(20)
```

Out[35]:

	Actual value	Predicted value
35	837	953.084940
329	3419	3043.168340
265	929	1177.631099
299	3600	1857.482226
190	1155	1107.036404
107	295	1002.222765

2.Features selection: Recursive Feature Elimination (RFE)

In [24]:
```
# first model with an arbitrary choice of n_features
# running RFE with number of features= X

rfe = RFE(lm, n_features_to_select=5)
rfe = rfe.fit(X_train, y_train)

# tuples of (feature name, whether selected, ranking)
# **note** that the 'rank' is > 1 for non-selected features

list(zip(X_train.columns,rfe.support_,rfe.ranking_))
```

Out[24]:
```
[('year_num', False, 5),
 ('province_num', False, 6),
 ('day_raindrop', False, 4),
 ('quant_rain', False, 3),
 ('humbidity_perc', False, 2),
 ('temp_max', True, 1),
 ('temp_min', True, 1),
 ('temp_avg', True, 1),
 ('dead', True, 1),
 ('dead_perc', True, 1)]
```

In [25]:
```
#r_sq = rfe.score(X_train,y_train)
#print('Coefficient of determination(R_squar_score):', r_sq)
```

In [26]:
```
# predict prices of X_test

predictions = rfe.predict(X_test)

# evaluate the model on test set

r2 = sklearn.metrics.r2_score(y_test, predictions)
print(r2)
```

```
0.6576483737049461
```

```
R squared of Training Set: 51.28
R squared of Test Set: 65.76
Mean Absolute Error (MAE): 618.6412
Root Mean Squared Error (RMSE): 940.3568
```

In [27]:
```
##----Result of training----##
# 1 = 0.3624
# 2 = 0.5066
# 3 = 0.5078
# 4 = 0.5103
# 5 = 0.5127
# 6 = 0.5138
# 7 = 0.5138
# 8 = 0.5140
# 9 = 0.5242
# 10 = 0.5246

##----Result of testing----##
# 1 = 0.5754
# 2 = 0.6540
# 3 = 0.6500
# 4 = 0.6537
# 5 = 0.6576 ***
# 6 = 0.6539
# 7 = 0.6529
# 8 = 0.6518
# 9 = 0.6401
# 10 = 0.6425

# Result generated highest score at 0.6576 in 5 Features
#[('year', False, 5),
 #('province_num', False, 6),
 #('day_raindrop', False, 4),
 #('quant_rain', False, 3),
 #('humbidity_perc', False, 2),
 #('temp_max', True, 1),
 #('temp_min', True, 1),
 #('temp_avg', True, 1),
 #('dead', True, 1),
 #('dead_perc', True, 1)]
```

3.GridSearchCV

In [22]:
```
# step-1: create a cross-validation scheme
folds = KFold(n_splits = 5, shuffle = False, random_state = 80)

# step-2: specify range of hyperparameters to tune
hyper_params = [{'n_features_to_select': list(range(1, 11))}]


# step-3: perform grid search
# 3.1 specify model
lm = LinearRegression()
lm.fit(X_train, y_train)
# Optional
rfe = RFE(lm)   # Recursive Feature Elimination
#sfs = SFS(lm)   # Sequential Feature Selector

# 3.2 call GridSearchCV()
model_cv = GridSearchCV(estimator = rfe,
                        param_grid = hyper_params,
                        scoring= 'r2',
                        cv = folds,
                        verbose = 1,
                        return_train_score=True)

# fit the model
model_cv.fit(X_train, y_train)
```

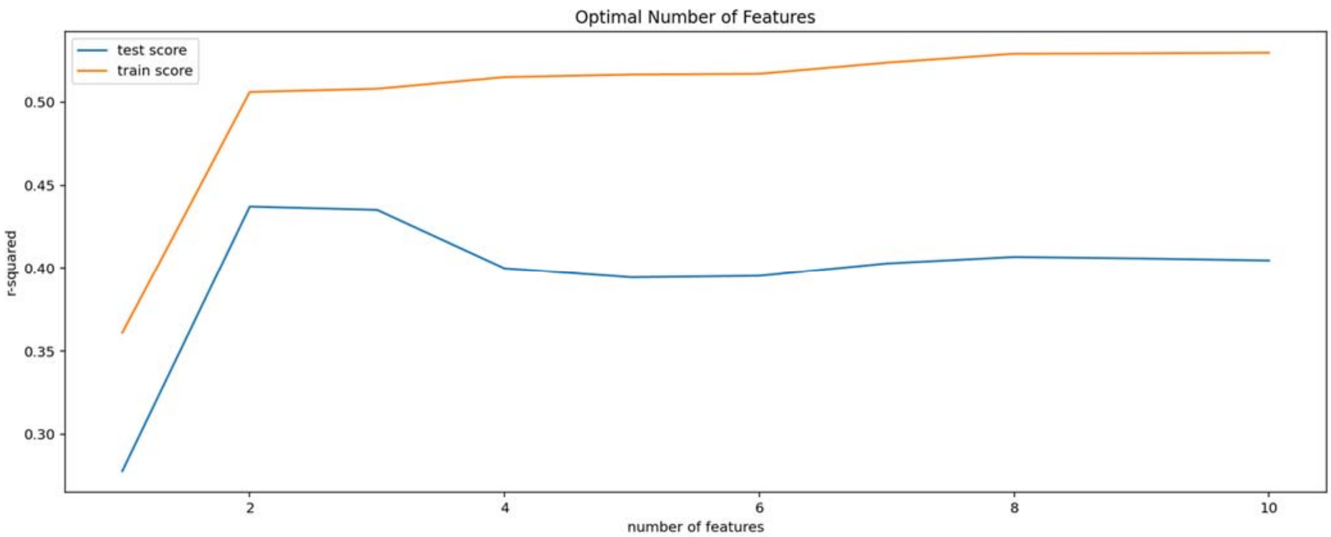
```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  50 out of  50 | elapsed:    0.5s finished
```

Out[22]:
```
GridSearchCV(cv=KFold(n_splits=5, random_state=80, shuffle=False),
             estimator=RFE(estimator=LinearRegression()),
             param_grid=[{'n_features_to_select': [1, 2, 3, 4, 5, 6, 7, 8, 9,
                                                   10]}],
             return_train_score=True, scoring='r2', verbose=1)
```

In [23]:
```
model_cv.best_params_
```

Out[23]:
```
{'n_features_to_select': 2}
```




```
R squared of Training Set: 50.66
R squared of Test Set: 65.40
Mean Absolute Error (MAE): 613.4758
Root Mean Squared Error (RMSE): 945.3093
```

4.K-folds Cross Validation(+Repeats)

Repeated k-Fold Cross-Validation in Python

The scikit-learn Python machine learning library provides an implementation of repeated k-fold cross-validation via the RepeatedKFold class.

The main parameters are the number of folds (n_splits), which is the "k" in k-fold cross-validation, and the number of repeats (n_repeats).

A good default for k is k=10.

A good default for the number of repeats depends on how noisy the estimate of model performance is on the dataset. A value of 3, 5, or 10 repeats is probably a good start. More repeats than 10 are probably not required.

```
from sklearn.model_selection import RepeatedKFold

kf = RepeatedKFold(n_splits=5, n_repeats=3, random_state=0)

for train_index, test_index in kf.split(X):
      print("Train:", train_index, "Validation:",test_index)

      X_train, X_test = X[train_index], X[test_index]
      y_train, y_test = y[train_index], y[test_index]
```

```
from sklearn import metrics
from sklearn.metrics import mean_squared_error

print('R squared of Training Set: {:.2f}'.format(lm.score(X_train,y_train)*100))
print('R squared of Test Set: {:.2f}'.format(lm.score(X_test,y_test)*100))
print('Mean Absolute Error (MAE): {:.4f}'.format(metrics.mean_absolute_error(y_test, predictions)))
print('Root Mean Squared Error (RMSE): {:.4f}'.format(np.sqrt(metrics.mean_squared_error(y_test, predictions))))
```

```
R squared of Training Set: 55.25
R squared of Test Set: 57.07
Mean Absolute Error (MAE): 676.5037
Root Mean Squared Error (RMSE): 998.2440
```

5.Polynomial regression (use degree = 2)

Train Test Split

This step we will separate data to train (training set) and test (testing set)

- training set use for train model
- testing set use for test model or call that Evaluation

In [18]:
```
from sklearn.model_selection import train_test_split

# Train dataset X0% and Test dataset X0%.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=17)
```

K-folds Cross varidation (optional)

from sklearn.model_selection import RepeatedKFold

kf = RepeatedKFold(n_splits=5, n_repeats=2, random_state=0)

for train_index, test_index in kf.split(X): print("Train:", train_index, "Validation:",test_index)

```
X_train, X_test = X[train_index], X[test_index]
y_train, y_test = y[train_index], y[test_index]
```

In [19]:
```
X_train, y_train = np.array(X_train), np.array(y_train)
X_test, y_test = np.array(X_test), np.array(y_test)
```

Creating and Training the Model

In [20]:
```
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures

X_train_poly = PolynomialFeatures(degree=2, include_bias=False).fit_transform(X_train)
X_test_poly = PolynomialFeatures(degree=2, include_bias=False).fit_transform(X_test)
```

Prediction

Actual value and the predicted value

In [26]:
```
#Actual value and the predicted value
mlr_diff = pd.DataFrame({'Actual value': y
mlr_diff.head(20)
```

Out[26]:

	Actual value	Predicted value
0	837	670.859924
1	3419	4024.796426
2	929	1467.842953
3	3600	2796.975016
4	1155	853.749229
5	295	442.073246
6	1542	1350.976802
7	7314	7472.123127
8	271	-244.214709
9	1557	845.457907
10	368	221.065791
11	7194	9131.088890
12	2092	2635.773511
13	1491	1769.509950
14	3211	3225.647126
15	580	551.858753
16	1950	1488.552027

```
R squared of Training Set: 83.42
R squared of Test Set: 78.01
Mean Absolute Error (MAE): 544.9456
Root Mean Squared Error (RMSE): 753.6509
```

Model Comparison

Model	Train R2 score	Test R2 score	RMSE	MAE
Linear Regression (original)	52.46	64.26	960.83	621.12
Recursive Feature Elimination (RFE)	51.28	65.76	940.36	618.64
Grid Search CV	50.66	65.4	945.31	613.48
Repeat K-folds Cross Validation	55.25	57.07	998.24	676.5
Polynomial Regression	83.42	78.01	753.65	544.95
Polynomial Regression + K-folds	83.6	72.5	713.25	536.25

Conclusion

We can used building up model of Polynomial regression or Polynomial regression with K-folds cross validation to deploy in unseen data set in 2021 to predict amount of patient of each provice…