FUTURE SEA LEVEL PREDICTION BASED ON LINEAR REGRESSION

BIG DATA: FROM BEGINNING TO ABYSS

第七組
11020107 電資二 蘇伯勳
11147106 財金一甲 楊寧寧
11014260 心理二乙 彭靖童

WORK ALLOCATION:

- 蘇伯勳: Whole project
- 楊寧寧: PPT layout, Topic Ideation
- 彭靖童: PPT layout, Topic Ideation

1. ABSTRACT

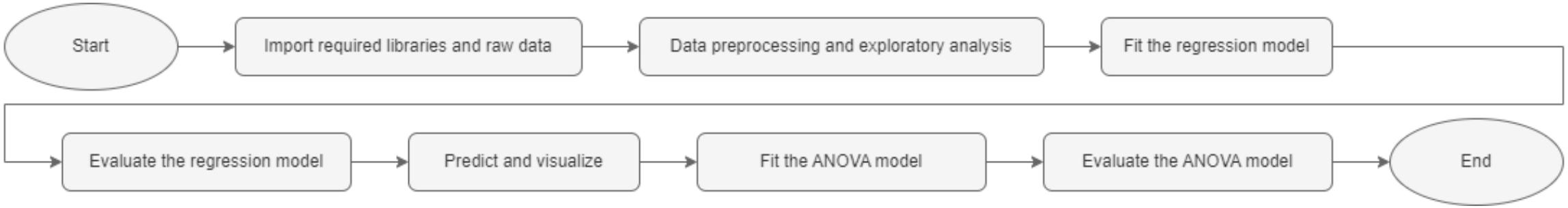
- Due to the raise of sea level caused by global warming, the sea level around Taiwan is correspondingly increasing.
- As Taiwanese, we hope to find out will Taiwan be submerged in the coming future.
- The data is from Central Weather Bureau, with missing data.
- The method we use is Unary Linear Regression and One-way ANOVA.
- The programming language we used is Python.
- We predicted the future 50 years, and the result tell us we have enough time to prevent and solving the problem.
- We visualized the relationship between sea level (independent variable) and time (dependent variable).

2. LITERATURE / INTRODUCTION

- The issue of rising sea levels has been studied since a long time ago. However, there has been comparatively less research conducted specifically on Taiwan in relation to the issue of rising sea levels. Hence, to gain a better understanding of our homeland and to maintain a constant sense of urgency, we have decided to utilize the data provided by the Central Weather Bureau to forecast the sea level rise in Taiwan for the next 50 years.
- We do research on the following stations: 成功, 竹圍, 高雄, 基隆, 將軍, 塭港, 蟳廣嘴, 蘇澳, and 蘭嶼.
- Every month in the station is a single linear regression model from sklearn, hence we have 9 stations * 12 months = 108 unary simple linear regression model. Then, We classify there locations by ‘north’, ‘south’ and ’middle’, put them into an one-way ANOVA model from scipy. Result shows that the p-value approaches zero infinitely, belongs to Alternative Hypothesis.

3. METHOD / DATASET / VARIABLES

A. The procedure of the whole project:



B. The raw data:

- Source: https://opendata.cwb.gov.tw/dataset/forecast/C-B0048-001
- Preprocessing:
  1. Filter the useless labels
  2. Impute the missing data


```
{
 "cwbdata": {
  "@xmlns": "urn:cwb.gov.tw:cwbdata-0.1",
  "@xmlns:xsi": "http://www.w3.org/2001/XMLSchema-instance",
  "@xsi:schemaLocation": "urn:cwb.gov.tw:cwbdata-0.1 ../../XSD/CWB-data-0.2.xsd",
  "Identifier": "3bb7baa6-1807-487d-91d8-06cd242068ce",
  "DatasetID": "C-MMC-0001",
  "DatasetName": "海象氣候統計",
  "DataItemID": "C-MMC-0001-00004",
  "Sender": "weather@cwb.gov.tw",
  "Sent": "2023-04-27T18:02:53+08:00",
  "Status": "Actual",
  "Scope": "Public",
  "MsgType": "Issue",
  "Resources": {
   "Resource": {
    "Metadata": {
     "ResourceID": "C-MMC-0001-00004",
     "ResourceName": "中央氣象局海象觀測_全臺_海平面統計資料",
     "ResourceDescription": "臺灣各地潮位觀測月平均海平面，包含：測站名稱、測站代碼、西元年月、平均海平面、最高高潮位、最低低潮位",
     "Language": "zh",
     "Homepage": "https://www.cwb.gov.tw/V8/C/C/MMC_STAT/sta_sea.html",
     "Sources": {
      "Source": {
       "Title": "中央氣象局所屬海象觀測站列表",
       "Path": "https://oceanapi.cwb.gov.tw/restapi/v2/static/station/station_info.html"
      }
     },
     "Temporal": {
      "Description": "資料更新頻率為每年"
     },
     "Spatial": {
      "Description": "中央氣象局所屬海象觀測站",
      "CoordinateReferenceSystem": {
       "EPSGCode": "EPSG:4326",
       "Name": "WGS84",
       "Datum": "World Geodetic System 1984(EPSG:6326)",
       "CoordinateFormat": "decimal degrees"
```

```
"Data": {
 "SeaSurfaceObs": {
  "Location": [
   {
    "Station": {
     "StationID": "1366",
     "StationName": "塭港",
     "StationNameEN": "Wengang",
     "StationLatitude": "23.47",
     "StationLongitude": "120.12",
     "StationAttribute": "潮位站",
     "County": {
      "CountyName": "嘉義縣",
      "CountyNameEN": "Chiayi County"
     },
     "Town": {
      "TownName": "東石鄉",
      "TownNameEN": "Dongshi Township"
     }
    },
    "StationObsStatistics": {
     "StartYear": "1963",
     "EndYear": "2023",
     "DataYear": [
      "1963",
      "1964",
      "1965",
      "1966",
      "1967",
      "1968",
      "1969",
      "1970",
      "1971",
      "1972",
      "1973",
      "1974",
      "1975",
```


time	level
196301	-
196302	-
196303	-0.060
196304	0.012
196305	0.084
196306	0.112
196307	0.149
196308	0.171
196309	0.187
196310	0.132
196311	0.098
196312	-0.026
196401	-0.111
196402	-0.150
196403	-0.108
196404	-
196405	0.084
196406	0.144
196407	-
196408	0.226
196409	0.244
196410	0.128
196411	0.003
196412	-0.162
196501	-
196502	-
196503	-
196504	-
196505	-
196506	-
196507	-
196508	-
196509	-
196510	-
196511	-
196512	-
196601	-

time	level
196301	-0.149
196302	-0.13
196303	-0.06
196304	0.012
196305	0.084
196306	0.112
196307	0.149
196308	0.171
196309	0.187
196310	0.132
196311	0.098
196312	-0.026
196401	-0.111
196402	-0.15
196403	-0.108
196404	-0.011999999999999997
196405	0.084
196406	0.144
196407	0.185
196408	0.226
196409	0.244
196410	0.128
196411	0.003
196412	-0.162
196501	-0.149
196502	-0.13
196503	-0.061
196504	-0.011999999999999997
196505	0.064
196506	0.10450000000000001
196507	0.174
196508	0.202
196509	0.166
196510	0.0945
196511	-0.004
196512	-0.10075
196601	-0.149

C. Clear the variables:

- Independent variable (x): Time (unit: month)
- Dependent variable (y): Relative sea level (unit: m)

```
def outputFile(self, stationName, statistics): # wirte time and MeanTideLevel to txt
    self.stationName = stationName
    # generate new folder "filtered" to save filtered data
    newFolderPath = "./filteredData"
    if not os.path.exists(newFolderPath):
        os.makedirs(newFolderPath)
    # write file
    file_path = os.path.join(newFolderPath, stationName + "_filtered.txt")
    with open(file_path, 'w' ) as f:
        f.write('time\tlevel\n')
        count = 0
        for y in statistics["DataYear"]:
                for i in range(12):
                    f.write(y)
                    if int(statistics["Monthly"][count]["DataMonth"]) < 10:
                        f.write('0')
                    f.write(statistics["Monthly"][count]["DataMonth"])
                    f.write('\t' + statistics["Monthly"][count]["MeanTideLevel"] + '\n')
                    count += 1
    self.filtered.append(file_path)
```


```
 "2021",
 "2022",
 "2023"
],
"Monthly": [
 {
  "DataMonth": "1",
  "HighestHighWaterLevel": "-",
  "MeanTideLevel": "-",
  "LowestLowWaterLevel": "-"
 },
 {
  "DataMonth": "2",
  "HighestHighWaterLevel": "-",
  "MeanTideLevel": "-",
  "LowestLowWaterLevel": "-"
 },
 {
  "DataMonth": "3",
  "HighestHighWaterLevel": "-",
  "MeanTideLevel": "-0.060",
  "LowestLowWaterLevel": "-"
 },
 {
  "DataMonth": "4",
  "HighestHighWaterLevel": "-",
  "MeanTideLevel": "0.012",
  "LowestLowWaterLevel": "-"
 },
 {
  "DataMonth": "5",
  "HighestHighWaterLevel": "-",
  "MeanTideLevel": "0.084",
  "LowestLowWaterLevel": "-"
 },
```

D. Methods of data imputation:

- Nearest neighbor imputation
- Mean imputation
- Median imputation
- Mode imputation

```python
def fill_in_with_Nearest_value_imputation(self):
    for i in range(len(self.data)):
        if self.data[i][1] is None:
            if i == 0 or i == len(self.data) - 1: # skip head and tail
                continue
            elif self.data[i-1][1] is not None and self.data[i+1][1] is not None: # use
                self.data[i][1] = (self.data[i-1][1] + self.data[i+1][1]) / 2
            # else the data is still None
```

```python
def fill_in_with_three_imputation_mode(self, imputationMode):
    year = []
    for i in range(12):
        year.append([])
    for row in self.data:
        if row[1] == None:
            continue
        month = int(row[0][-2:])
        year[month-1].append(float(row[1]))

    # get each month's median value and append in medianMonthData
    medianMonthData = [] # store each value of month
    for month in year:
        if imputationMode == 'mean':
            median = statistics.mean(month)
        elif imputationMode == 'median':
            median = statistics.median(month)
        elif imputationMode == 'mode':
            median = statistics.mode(month)
        medianMonthData.append(median)

    # replace None to the median value of each month
    for row in self.data:
        if row[1] == None:
            row[1] = medianMonthData[int(row[0][4:])-1]
```

`statistics.`**`mean`**`(`*`data`*`)`

Return the sample arithmetic mean of *data* which can be a sequence or iterable.

The arithmetic mean is the sum of the data divided by the number of data points. It is commonly called “the average”, although it is only one of many different mathematical averages. It is a measure of the central location of the data.

`statistics.`**`median`**`(`*`data`*`)`

Return the median (middle value) of numeric data, using the common “mean of middle two” method. If *data* is empty, `StatisticsError` is raised. *data* can be a sequence or iterable.

`statistics.`**`mode`**`(`*`data`*`)`

Return the single most common data point from discrete or nominal *data*. The mode (when it exists) is the most typical value and serves as a measure of central location.

If there are multiple modes with the same frequency, returns the first one encountered in the *data*. If the smallest or largest of those is desired instead, use `min(multimode(data))` or `max(multimode(data))`. If the input *data* is empty, `StatisticsError` is raised.

E. Train the linear regression model:

1. Split x, y of the train & test data (80% for training, 20% for testing)
2. Train model
3. Get training score

```
# train
x_train, x_test, y_train, y_test = train_test_split(self.monthX, self.monthY[i], test_size = 0.2, random_state = 42)
model = LinearRegression()
model.fit(x_train, y_train) # train
train_score = model.score(x_train, y_train)
test_score = model.score(x_test, y_test)

## add the single object to the set
self.models.append(model)
self.train_scores.append(train_score)
self.test_scores.append(test_score)
self.coefs.append(model.coef_[0][0])
self.intercepts.append(model.intercept_[0])
```

class `sklearn.linear_model.LinearRegression`(*, *fit_intercept=True*, *copy_X=True*, *n_jobs=None*, *positive=False*) [source]

Ordinary least squares Linear Regression.

LinearRegression fits a linear model with coefficients w = (w1, ..., wp) to minimize the residual sum of squares between the observed targets in the dataset, and the targets predicted by the linear approximation.

Parameters:

fit_intercept : *bool, default=True*
Whether to calculate the intercept for this model. If set to False, no intercept will be used in calculations (i.e. data is expected to be centered).

copy_X : *bool, default=True*
If True, X will be copied; else, it may be overwritten.

n_jobs : *int, default=None*
The number of jobs to use for the computation. This will only provide speedup in case of sufficiently large problems, that is if firstly `n_targets > 1` and secondly `X` is sparse or if `positive` is set to `True`. `None` means 1 unless in a `joblib.parallel_backend` context. `-1` means using all processors. See Glossary for more details.

positive : *bool, default=False*
When set to `True`, forces the coefficients to be positive. This option is only supported for dense arrays.

New in version 0.24.

Attributes:

coef_ : *array of shape (n_features,) or (n_targets, n_features)*
Estimated coefficients for the linear regression problem. If multiple targets are passed during the fit (y 2D), this is a 2D array of shape (n_targets, n_features), while if only one target is passed, this is a 1D array of length n_features.

rank_ : *int*
Rank of matrix `X`. Only available when `X` is dense.

singular_ : *array of shape (min(X, y),)*
Singular values of `X`. Only available when `X` is dense.

intercept_ : *float or array of shape (n_targets,)*
Independent term in the linear model. Set to 0.0 if `fit_intercept = False`.

n_features_in_ : *int*
Number of features seen during fit.

New in version 0.24.

feature_names_in_ : *ndarray of shape (`n_features_in_`,)*
Names of features seen during fit. Defined only when `X` has feature names that are all strings.

New in version 1.0.

`sklearn.model_selection.train_test_split`(*arrays, test_size=None, train_size=None, random_state=None, shuffle=True, stratify=None) [source]

Split arrays or matrices into random train and test subsets.

Quick utility that wraps input validation, `next(ShuffleSplit().split(X, y))`, and application to input data into a single call for splitting (and optionally subsampling) data into a one-liner.

Read more in the User Guide.

Parameters:

arrays : *sequence of indexables with same length / shape[0]
Allowed inputs are lists, numpy arrays, scipy-sparse matrices or pandas dataframes.

test_size : *float or int, default=None*
If float, should be between 0.0 and 1.0 and represent the proportion of the dataset to include in the test split. If int, represents the absolute number of test samples. If None, the value is set to the complement of the train size. If `train_size` is also None, it will be set to 0.25.

train_size : *float or int, default=None*
If float, should be between 0.0 and 1.0 and represent the proportion of the dataset to include in the train split. If int, represents the absolute number of train samples. If None, the value is automatically set to the complement of the test size.

random_state : *int, RandomState instance or None, default=None*
Controls the shuffling applied to the data before applying the split. Pass an int for reproducible output across multiple function calls. See Glossary.

shuffle : *bool, default=True*
Whether or not to shuffle the data before splitting. If shuffle=False then stratify must be None.

stratify : *array-like, default=None*
If not None, data is split in a stratified fashion, using this as the class labels. Read more in the User Guide.

Returns:

splitting : *list, length=2 * len(arrays)*
List containing train-test split of inputs.

New in version 0.16: If the input is sparse, the output will be a `scipy.sparse.csr_matrix`. Else, output type is the same as the input type.

`fit`(*X, y, sample_weight=None*)

Fit linear model.

Parameters:

X : *{array-like, sparse matrix} of shape (n_samples, n_features)*
Training data.

y : *array-like of shape (n_samples,) or (n_samples, n_targets)*
Target values. Will be cast to X's dtype if necessary.

sample_weight : *array-like of shape (n_samples,), default=None*
Individual weights for each sample.

New in version 0.17: parameter *sample_weight* support to LinearRegression.

Returns:

self : *object*
Fitted Estimator.

`score`(*X, y, sample_weight=None*) [source]

Return the coefficient of determination of the prediction.

The coefficient of determination R^2 is defined as $(1 - \frac{u}{v})$, where u is the residual sum of squares `((y_true - y_pred)** 2).sum()` and v is the total sum of squares `((y_true - y_true.mean()) ** 2).sum()`. The best possible score is 1.0 and it can be negative (because the model can be arbitrarily worse). A constant model that always predicts the expected value of `y`, disregarding the input features, would get a R^2 score of 0.0.

Parameters:

X : *array-like of shape (n_samples, n_features)*
Test samples. For some estimators this may be a precomputed kernel matrix or a list of generic objects instead with shape `(n_samples, n_samples_fitted)`, where `n_samples_fitted` is the number of samples used in the fitting for the estimator.

y : *array-like of shape (n_samples,) or (n_samples, n_outputs)*
True values for `X`.

sample_weight : *array-like of shape (n_samples,), default=None*
Sample weights.

Returns:

score : *float*
R^2 of `self.predict(X)` w.r.t. `y`.

F. Predict the coming 50 years (2024 ~ 2074):

predict(*X*)

Predict using the linear model.

Parameters:	**X :** ***array-like or sparse matrix, shape (n_samples, n_features)*** Samples.
Returns:	**C :** ***array, shape (n_samples,)*** Returns predicted values.

```
# predict
start_predict_year = int(self.monthX[len(self.monthX)-1][0]) + 1
end_predict_year = start_predict_year + 50
future_years = np.arange(start_predict_year, end_predict_year + 1)
self.x_future = future_years.reshape(-1, 1)
self.y_future.append(model.predict(self.x_future))
```

G. Get p-value using One-way ANOVA:

1. Classify by location: North / Middle / South
2. Process One-way ANOVA, get p-value

scipy.stats.f_oneway

scipy.stats.f_oneway(**samples, axis=0*) [source]

Perform one-way ANOVA.

The one-way ANOVA tests the null hypothesis that two or more groups have the same population mean. The test is applied to samples from two or more groups, possibly with differing sizes.

Parameters: **sample1, sample2, ... : *array_like***

The sample measurements for each group. There must be at least two arguments. If the arrays are multidimensional, then all the dimensions of the array must be the same except for *axis*.

axis : *int, optional*

Axis of the input arrays along which the test is applied. Default is 0.

Returns: **statistic : *float***

The computed F statistic of the test.

pvalue : *float*

The associated p-value from the F distribution.

Warns: `ConstantInputWarning`

Raised if all values within each of the input arrays are identical. In this case the F statistic is either infinite or isn't defined, so `np.inf` or `np.nan` is returned.

`DegenerateDataWarning`

Raised if the length of any input array is 0, or if all the input arrays have length 1. `np.nan` is returned for the F statistic and the p-value in these cases.

anova.py

```python
import numpy as np
from scipy.stats import f_oneway

class ANOVA:
    def __init__(self, loc, dataSet):
        self.stations = {
            'loc': loc,
            'level': dataSet
        }

    def classify(self):
        station_data = {}
        for loc, level in zip(self.stations['loc'], self.stations['level']):
            if loc not in station_data:
                station_data[loc] = []
            station_data[loc].extend(level)
        return station_data

    def one_way_ANOVA(self, station_data):
        f_statistic, p_value = f_oneway(*station_data.values())
        print('f_statistic: ')
        print(f_statistic)
        print('p_value: ')
        print('{:.2e}'.format(p_value))
        if p_value == 0:
            print('p_value is zero!')
        elif p_value < 0.05:
            print('Alternative Hypothesis!')
        else:
            print('Null Hypothesis!')

    def main_control(self):
        self.one_way_ANOVA(self.classify())
```

4. RESULTS / FORMULA

- First we will present every linear regression models of every stations, then we will show the result of ANOVA.
- Formula: y = coefficient * x + intercept
- Results show that the future sea level indeed increasing.
- P-value indicates that the alternative hypothesis is supported.

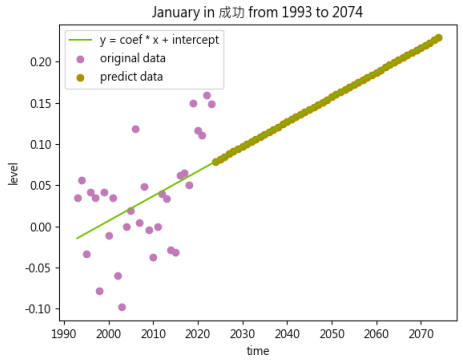
成功 STATION

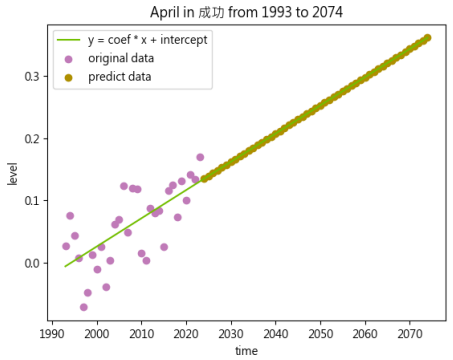
```
Month: January
Train score: 0.1968645770333155
Test score: 0.4035499745894827
Train MSE: 0.003017601482926549
Test MSE: 0.003094334042031771
Coefficient: 0.003005725312241567
Intercept: -6.004740940223503
----------------
Month: Faburary
Train score: 0.2667535464847097
Test score: 0.43601245555875023
Train MSE: 0.0028593849282922486
Test MSE: 0.0029712014836153697
Coefficient: 0.003564473377298078
Intercept: -7.1290882546279635
----------------
Month: March
Train score: 0.547504589406446
Test score: 0.502685047450532
Train MSE: 0.001552423768527757
Test MSE: 0.0025217927808156393
Coefficient: 0.00478983863101423
Intercept: -9.579239042388519
```

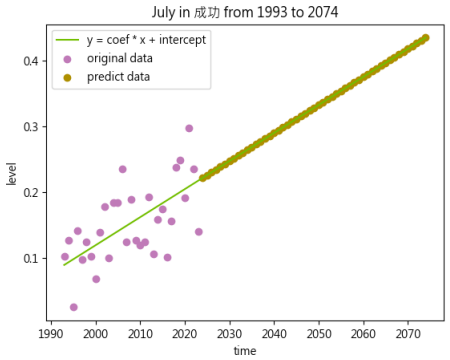
```
Month: April
Train score: 0.47334321325444706
Test score: 0.5222856577212169
Train MSE: 0.0018824386790785616
Test MSE: 0.0018021919801551464
Coefficient: 0.004545834305222758
Intercept: -9.066053860344793
----------------
Month: May
Train score: 0.21566493507447937
Test score: 0.11537570533652408
Train MSE: 0.002085429491631144
Test MSE: 0.004145963265732202
Coefficient: 0.002646461969083314
Intercept: -5.227746941198924
----------------
Month: June
Train score: 0.4279117506514739
Test score: 0.5573309655578564
Train MSE: 0.001684131464831736
Test MSE: 0.001387306685817823
Coefficient: 0.003922496236137323
Intercept: -7.757155049937447
```

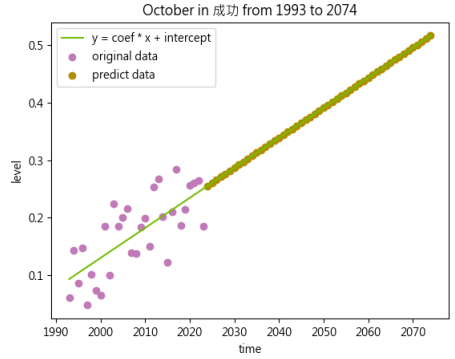
```
Month: July
Train score: 0.40350147548545456
Test score: -0.05455592240023077
Train MSE: 0.0022096034779504766
Test MSE: 0.0020111027086655497
Coefficient: 0.004272715706439918
Intercept: -8.42622678597935
----------------
Month: August
Train score: 0.33936205915358575
Test score: 0.40394996402102423
Train MSE: 0.0034504007117764778
Test MSE: 0.001005694546420363
Coefficient: 0.004652791619839269
Intercept: -9.138888314001573
----------------
Month: September
Train score: 0.30217933627684457
Test score: 0.3742573046881189
Train MSE: 0.002911942632017919
Test MSE: 0.0029440044491106086
Coefficient: 0.0039244682881316405
Intercept: -7.679988040458875
```

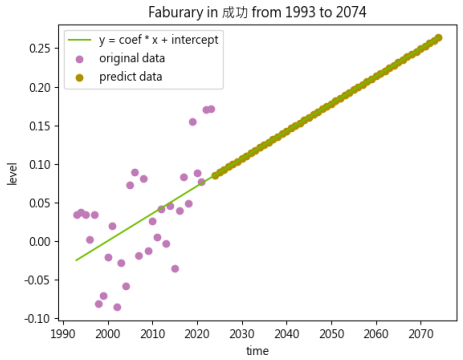
```
Month: October
Train score: 0.48329960446665876
Test score: 0.6970840545161028
Train MSE: 0.002384948861829122
Test MSE: 0.0009119006350679526
Coefficient: 0.005219841387646051
Intercept: -10.30970999173011
----------------
Month: November
Train score: 0.3128565529489705
Test score: 0.3665901336491537
Train MSE: 0.0017222630162499909
Test MSE: 0.0018994798485984907
Coefficient: 0.0030947645200279906
Intercept: -6.125404673551178
----------------
Month: December
Train score: 0.22956616135891472
Test score: -0.04718391690369761
Train MSE: 0.0017672989849940339
Test MSE: 0.003363768452098127
Coefficient: 0.00253612247927225
Intercept: -5.052478487669375
```

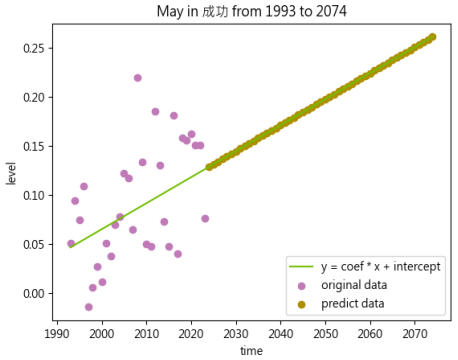

January in 成功 from 1993 to 2074
y = coef * x + intercept
original data
predict data
level
time

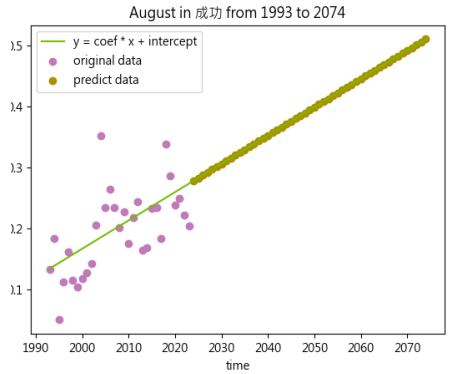

April in 成功 from 1993 to 2074
y = coef * x + intercept
original data
predict data
level
time

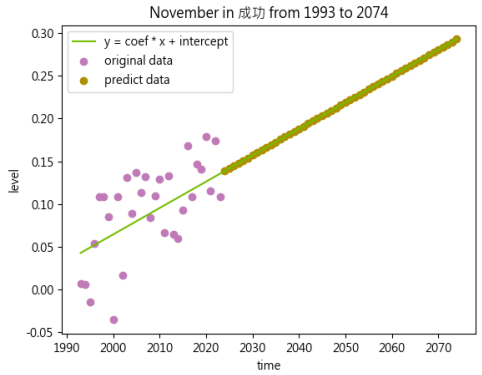

July in 成功 from 1993 to 2074
y = coef * x + intercept
original data
predict data
level
time

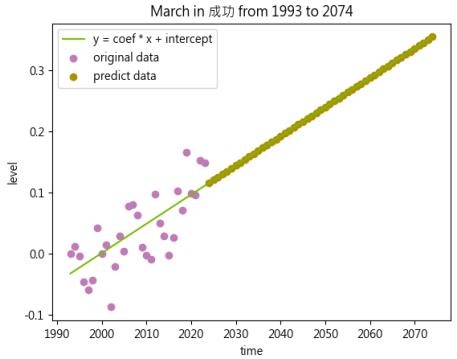

October in 成功 from 1993 to 2074
y = coef * x + intercept
original data
predict data
level
time

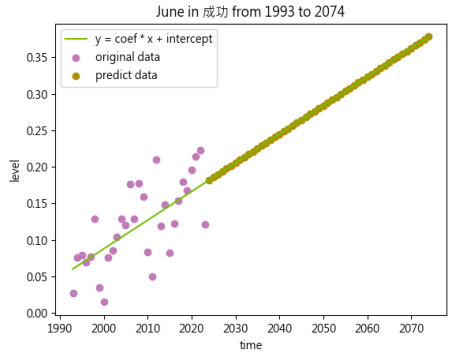

Faburary in 成功 from 1993 to 2074
y = coef * x + intercept
original data
predict data
level
time

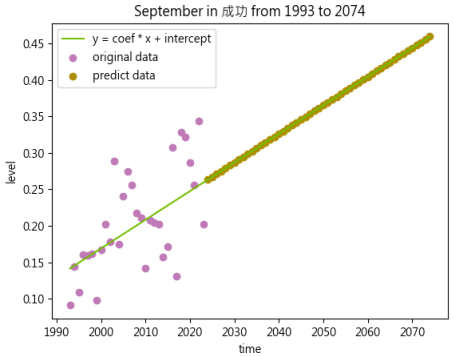

May in 成功 from 1993 to 2074
y = coef * x + intercept
original data
predict data
level
time

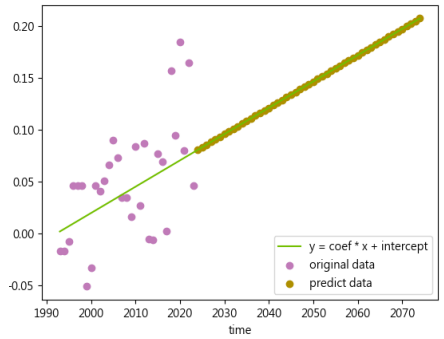

August in 成功 from 1993 to 2074
y = coef * x + intercept
original data
predict data
time

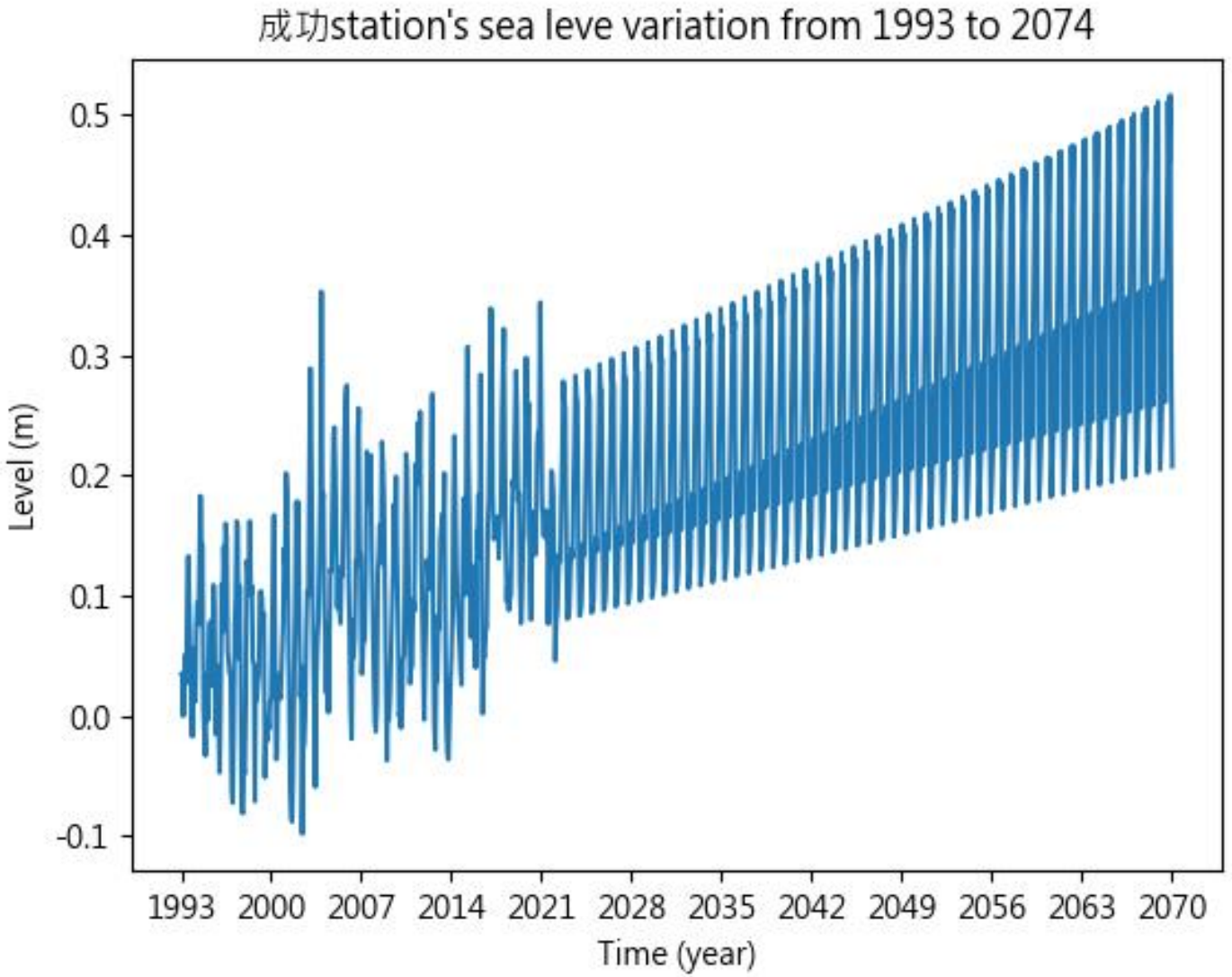

November in 成功 from 1993 to 2074
y = coef * x + intercept
original data
predict data
level
time


March in 成功 from 1993 to 2074
y = coef * x + intercept
original data
predict data
level
time


June in 成功 from 1993 to 2074
y = coef * x + intercept
original data
predict data
level
time


September in 成功 from 1993 to 2074
y = coef * x + intercept
original data
predict data
level
time


December in 成功 from 1993 to 2074
y = coef * x + intercept
original data
predict data
level
time


成功station's sea leve variation from 1993 to 2074
Level (m)
0.5
0.4
0.3
0.2
0.1
0.0
-0.1
1993
2000
2007
2014
2021
2028
2035
2042
2049
2056
2063
2070
Time (year)

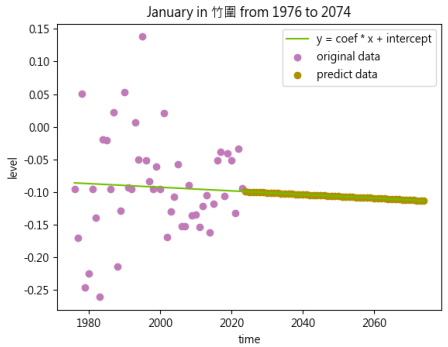
竹圍 STATION

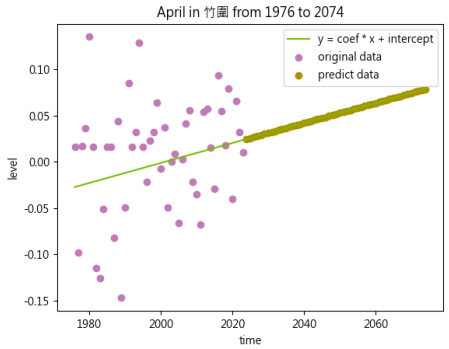
```
Month: January
Train score: 0.003474274504566943
Test score: -0.025508110505195525
Train MSE: 0.004543298805487349
Test MSE: 0.011263658076652713
Coefficient: -0.0002775993989400594
Intercept: 0.46247823010757816
----------------
Month: Faburary
Train score: 0.0007882301430295335
Test score: 0.006328728998919786
Train MSE: 0.002940971560126121
Test MSE: 0.010844938188418498
Coefficient: 0.00010624003989097377
Intercept: -0.2887601892125915
----------------
Month: March
Train score: 0.00437771109870011
Test score: 0.0205929833286419
Train MSE: 0.0031266876758063978
Test MSE: 0.014086037389240974
Coefficient: 0.00025862086392259003
Intercept: -0.5696205842814741
```

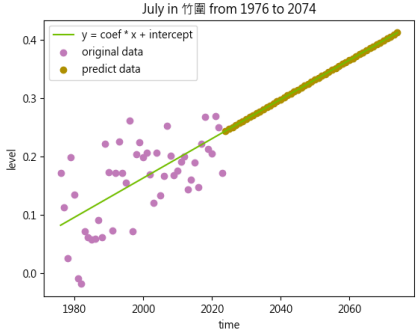
```
Month: April
Train score: 0.06716559147309265
Test score: -0.7018096887914254
Train MSE: 0.0033313822974228857
Test MSE: 0.00442002521421353
Coefficient: 0.0010802603694631898
Intercept: -2.1618025632645903
----------------
Month: May
Train score: 0.2720159890267366
Test score: 0.47418245523459956
Train MSE: 0.0031558656392707556
Test MSE: 0.0010397568711945503
Coefficient: 0.0023951935419748048
Intercept: -4.7280396382859005
----------------
Month: June
Train score: 0.4140955995175374
Test score: 0.3171892314254561
Train MSE: 0.0034948406274387667
Test MSE: 0.0018893100842150203
Coefficient: 0.0034665357182565204
Intercept: -6.814568069027553
```

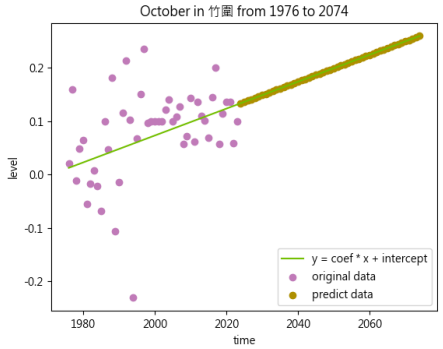
```
Month: July
Train score: 0.40336643194829025
Test score: -0.07036560124887847
Train MSE: 0.003461240319354765
Test MSE: 0.002012940253364654
Coefficient: 0.0033740924298117634
Intercept: -6.585465444107151
----------------
Month: August
Train score: 0.3995002559648103
Test score: -0.4858722121589836
Train MSE: 0.004919552151294893
Test MSE: 0.0025595040377765785
Coefficient: 0.003990340589402611
Intercept: -7.786507390274551
----------------
Month: September
Train score: 0.3373400024947042
Test score: -0.32618344728614046
Train MSE: 0.005995944069116928
Test MSE: 0.0034374144480277848
Coefficient: 0.0038535623919759834
Intercept: -7.543545940318519
```

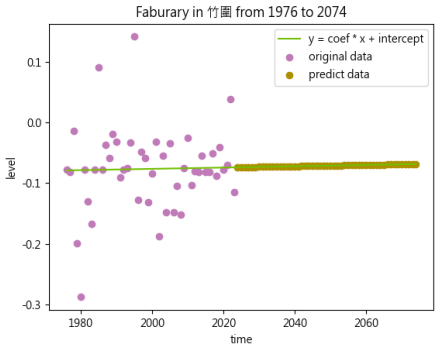
```
Month: October
Train score: 0.1603085393228093
Test score: -1.404037363559782
Train MSE: 0.00684631049526622
Test MSE: 0.0025450642053996067
Coefficient: 0.0025216907169256985
Intercept: -4.970384346498925
----------------
Month: November
Train score: 0.16574576095771376
Test score: 0.20572191577666998
Train MSE: 0.005773673113189401
Test MSE: 0.009918984429685158
Coefficient: 0.002362344133769528
Intercept: -4.743902580447358
----------------
Month: December
Train score: 0.07597250376906939
Test score: -0.09107819989535537
Train MSE: 0.008521638813353473
Test MSE: 0.0058774309579782994
Coefficient: 0.0018462580312591593
Intercept: -3.7761293104319598
```

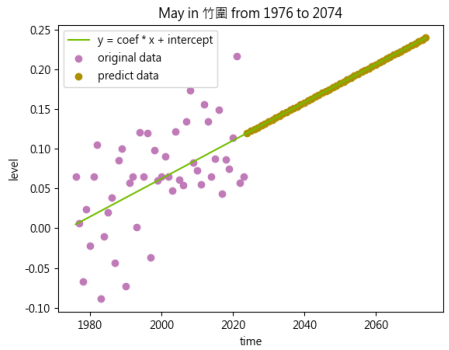

January in 竹圍 from 1976 to 2074
y = coef * x + intercept
original data
predict data
level
time

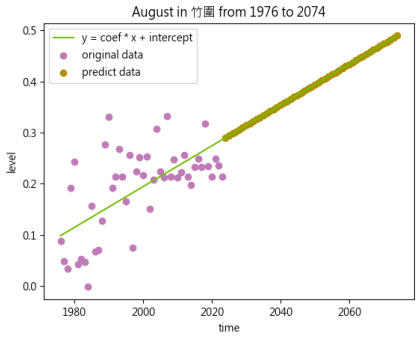

April in 竹圍 from 1976 to 2074
y = coef * x + intercept
original data
predict data
level
time

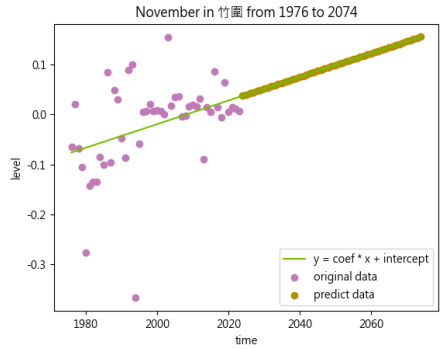

July in 竹圍 from 1976 to 2074
y = coef * x + intercept
original data
predict data
level
time

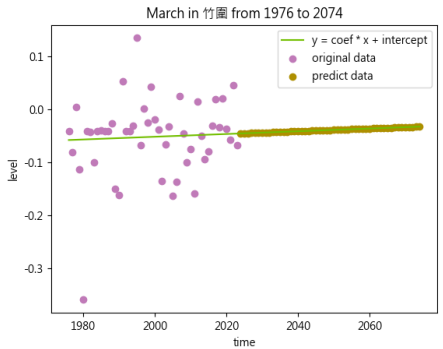

October in 竹圍 from 1976 to 2074
y = coef * x + intercept
original data
predict data
level
time

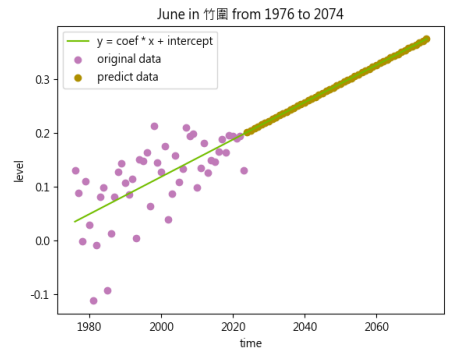

Faburary in 竹圍 from 1976 to 2074
y = coef * x + intercept
original data
predict data
level
time

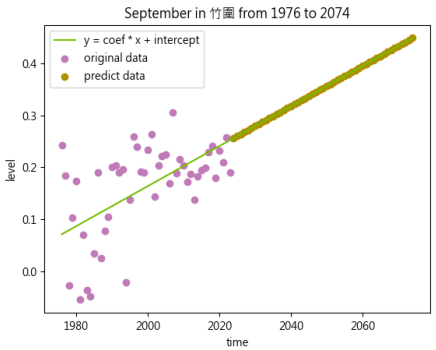

May in 竹圍 from 1976 to 2074
y = coef * x + intercept
original data
predict data
level
time

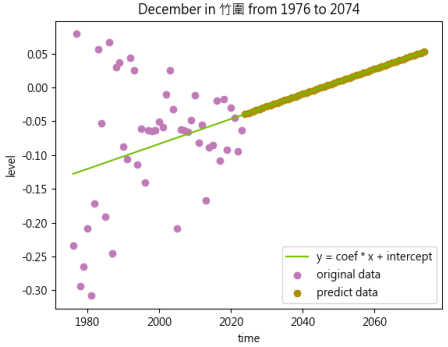

August in 竹圍 from 1976 to 2074
y = coef * x + intercept
original data
predict data
level
time

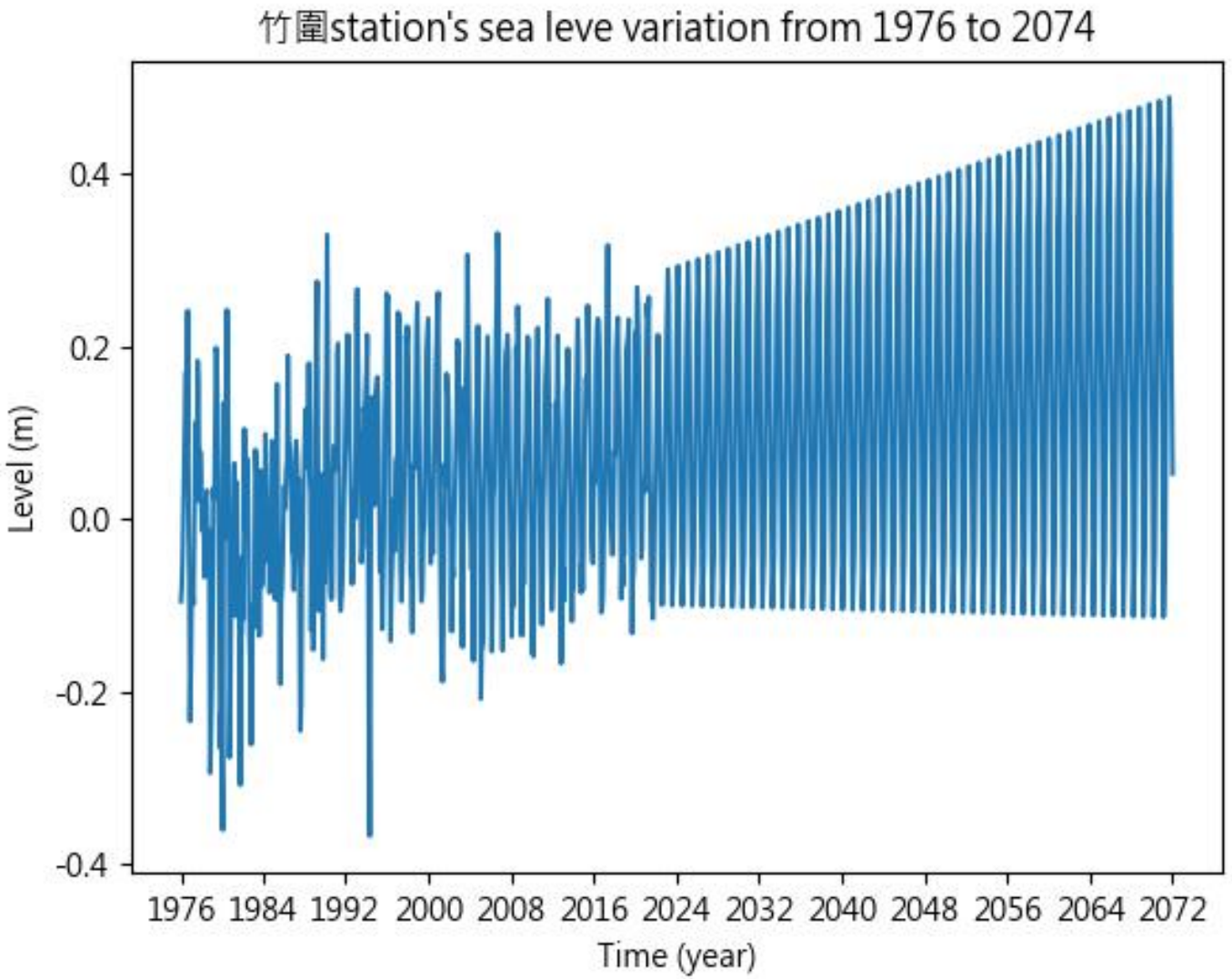

November in 竹圍 from 1976 to 2074
y = coef * x + intercept
original data
predict data
level
time


March in 竹圍 from 1976 to 2074
y = coef * x + intercept
original data
predict data
level
time


June in 竹圍 from 1976 to 2074
y = coef * x + intercept
original data
predict data
level
time


September in 竹圍 from 1976 to 2074
y = coef * x + intercept
original data
predict data
level
time


December in 竹圍 from 1976 to 2074
y = coef * x + intercept
original data
predict data
level
time


竹圍station's sea leve variation from 1976 to 2074
0.4
0.2
0.0
-0.2
-0.4
Level (m)
1976 1984 1992 2000 2008 2016 2024 2032 2040 2048 2056 2064 2072
Time (year)

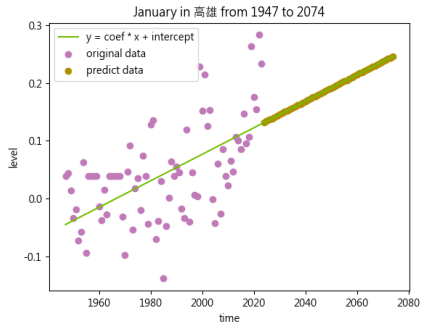
高雄 STATION

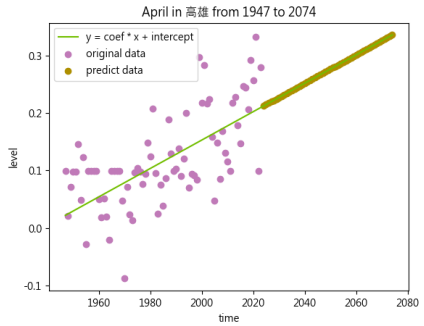
```
Month: January
Train score: 0.3364676390708502
Test score: -0.0353680363851665
Train MSE: 0.0050147322142361245
Test MSE: 0.004611751676411099
Coefficient: 0.002290584646027317
Intercept: -4.504401245192457
----------------
Month: Faburary
Train score: 0.3958096930147347
Test score: -0.06646250676675569
Train MSE: 0.004905803815811263
Test MSE: 0.004587750903474734
Coefficient: 0.002575095263655197
Intercept: -5.053662879779265
----------------
Month: March
Train score: 0.43152019090894966
Test score: -0.07445048647464203
Train MSE: 0.0037051683699066635
Test MSE: 0.006179899235352905
Coefficient: 0.002408956114396235
Intercept: -4.693890390237195
```

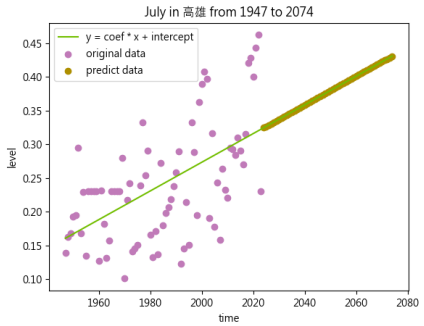
```
Month: April
Train score: 0.4359362481198584
Test score: 0.19975071786524623
Train MSE: 0.0038300331980175724
Test MSE: 0.003989030105226181
Coefficient: 0.0024713288559045725
Intercept: -4.78970743965896
----------------
Month: May
Train score: 0.33202354500086717
Test score: 0.09101135350050804
Train MSE: 0.0037182212848138013
Test MSE: 0.0036692889040188157
Coefficient: 0.0019527801683028212
Intercept: -3.7231894449200857
----------------
Month: June
Train score: 0.3200199391318168
Test score: 0.21407735018057683
Train MSE: 0.004088698359795804
Test MSE: 0.003774491766089007
Coefficient: 0.001992575756231508
Intercept: -3.7550489094030453
```

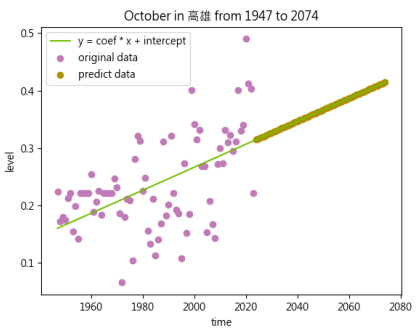
```
Month: July
Train score: 0.31587081251057847
Test score: 0.31899370139469085
Train MSE: 0.004745059923425738
Test MSE: 0.005034073044354355
Coefficient: 0.0021261231462956966
Intercept: -3.9789017054301388
----------------
Month: August
Train score: 0.3929045707596587
Test score: 0.320627971423714
Train MSE: 0.0035733409365483215
Test MSE: 0.003980938965812699
Coefficient: 0.002184410030741659
Intercept: -4.056779990861787
----------------
Month: September
Train score: 0.41787167612805975
Test score: 0.15638393575085963
Train MSE: 0.004051472021612253
Test MSE: 0.003058688218947305
Coefficient: 0.0024496294809406155
Intercept: -4.599271034525591
```

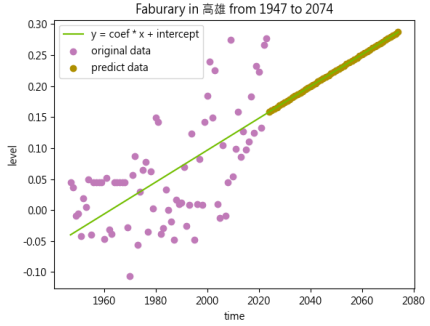
```
Month: October
Train score: 0.29013140396483694
Test score: 0.15445875263920505
Train MSE: 0.004758823618557927
Test MSE: 0.004019065631122201
Coefficient: 0.0020032726780731683
Intercept: -3.739890084151411
----------------
Month: November
Train score: 0.3038738915027913
Test score: -0.00246844252398426
Train MSE: 0.0035218233980565 31
Test MSE: 0.006129088141699286
Coefficient: 0.0017810183562749803
Intercept: -3.3707225376729495
----------------
Month: December
Train score: 0.38701781329693796
Test score: 0.14912817806764223
Train MSE: 0.003653063140892114
Test MSE: 0.005720337306124529
Coefficient: 0.0021814839171882703
Intercept: -4.247184434428886
```

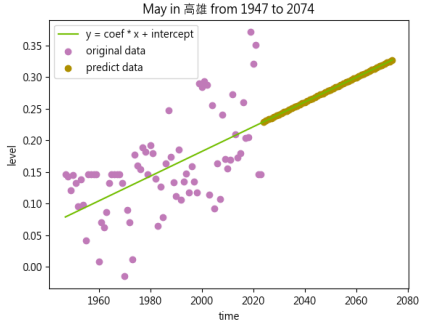

January in 高雄 from 1947 to 2074
y = coef * x + intercept
original data
predict data
level
time

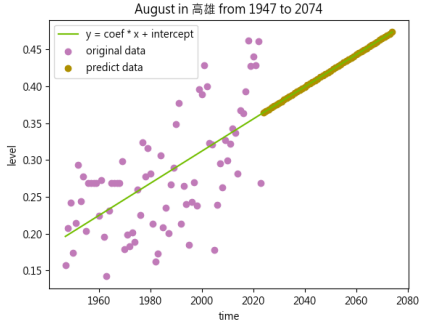

April in 高雄 from 1947 to 2074
y = coef * x + intercept
original data
predict data
level
time

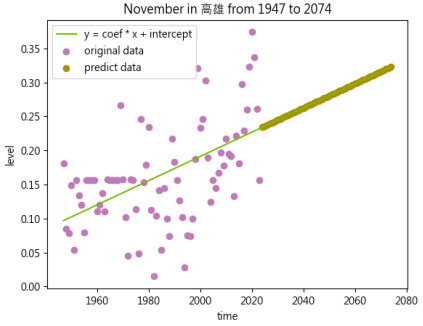

July in 高雄 from 1947 to 2074
y = coef * x + intercept
original data
predict data
level
time

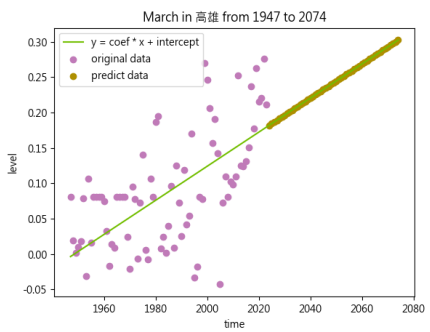

October in 高雄 from 1947 to 2074
y = coef * x + intercept
original data
predict data
level
time

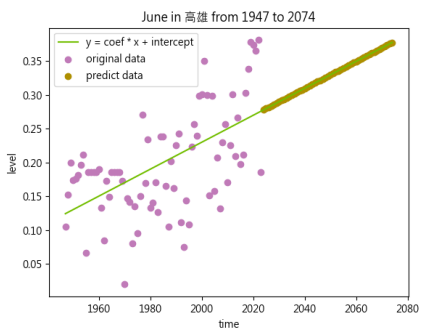

Faburary in 高雄 from 1947 to 2074
y = coef * x + intercept
original data
predict data
level
time

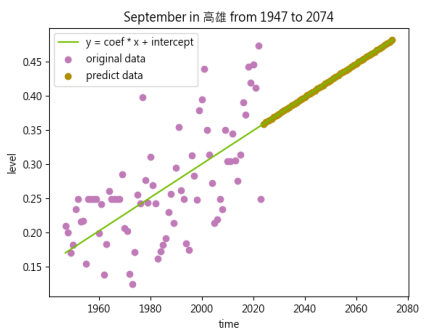

May in 高雄 from 1947 to 2074
y = coef * x + intercept
original data
predict data
level
time

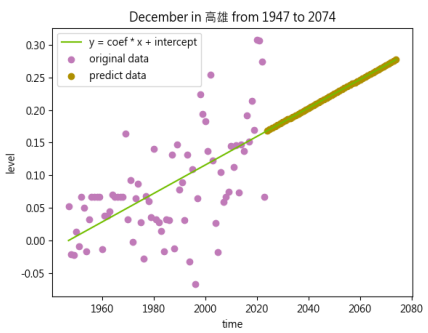

August in 高雄 from 1947 to 2074
y = coef * x + intercept
original data
predict data
level
time

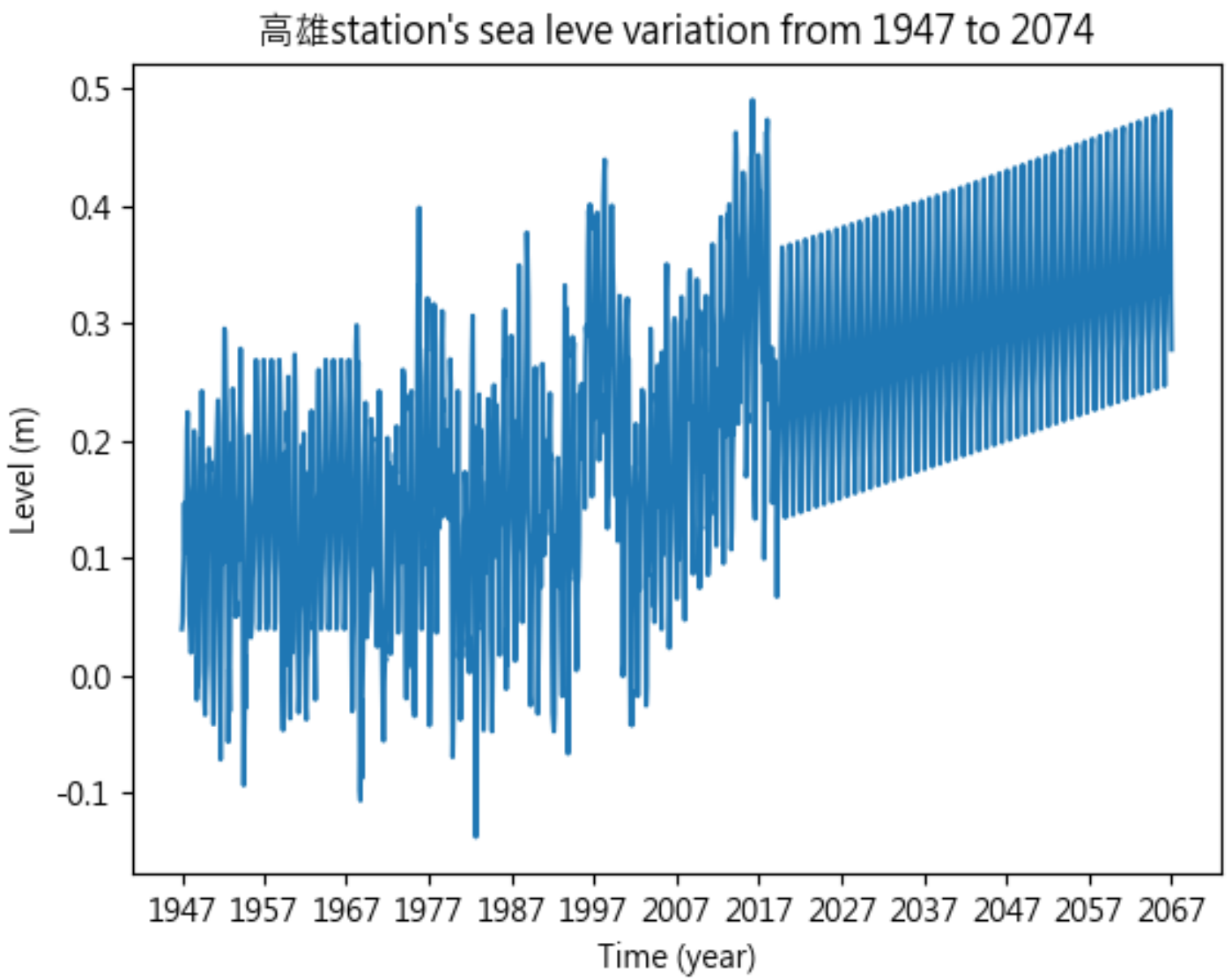

November in 高雄 from 1947 to 2074
y = coef * x + intercept
original data
predict data
level
time


March in 高雄 from 1947 to 2074
y = coef * x + intercept
original data
predict data
level
time


June in 高雄 from 1947 to 2074
y = coef * x + intercept
original data
predict data
level
time


September in 高雄 from 1947 to 2074
y = coef * x + intercept
original data
predict data
level
time


December in 高雄 from 1947 to 2074
y = coef * x + intercept
original data
predict data
level
time


高雄station's sea leve variation from 1947 to 2074
0.5
0.4
0.3
0.2
0.1
0.0
-0.1
Level (m)
1947 1957 1967 1977 1987 1997 2007 2017 2027 2037 2047 2057 2067
Time (year)

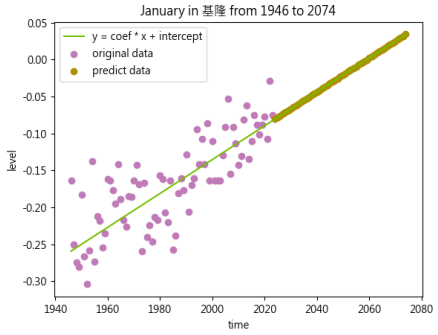
基隆 STATION

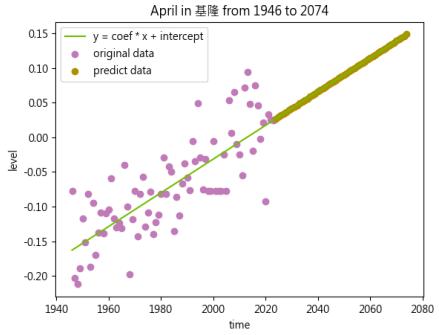
```
Month: January
Train score: 0.6335801451434375
Test score: 0.3338788843610063
Train MSE: 0.0014823856482925773
Test MSE: 0.00171325310128106
Coefficient: 0.0022911090760017456
Intercept: -4.717524239910626
----------------
Month: Faburary
Train score: 0.6457343012359247
Test score: 0.5949041232567026
Train MSE: 0.0015377741317148853
Test MSE: 0.00155625213747639
Coefficient: 0.00239586606423167 4
Intercept: -4.913949042885934
----------------
Month: March
Train score: 0.6374710068374392
Test score: 0.6281292471725702
Train MSE: 0.001610093113815114
Test MSE: 0.0024374370177482857
Coefficient: 0.002407898781544861
Intercept: -4.903543470095879
```

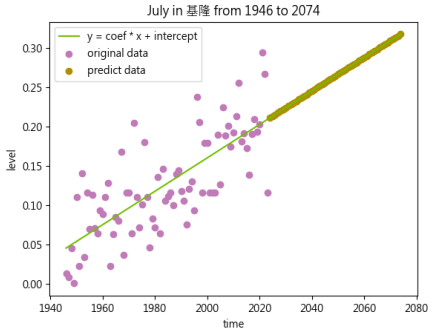
```
Month: April
Train score: 0.5988584414106709
Test score: 0.6842457418593466
Train MSE: 0.0019339953909292694
Test MSE: 0.0015363997827739467
Coefficient: 0.002431615264172646
Intercept: -4.894604176555861
----------------
Month: May
Train score: 0.611442551789862
Test score: 0.5704650812274701
Train MSE: 0.0018391876565547222
Test MSE: 0.0019779546090826538
Coefficient: 0.002434541133449367
Intercept: -4.826831334717422
----------------
Month: June
Train score: 0.6123062747829132
Test score: 0.5089258987042895
Train MSE: 0.0014446413482400204
Test MSE: 0.0022497979677755
Coefficient: 0.002161592744781836
Intercept: -4.202041250121867
```

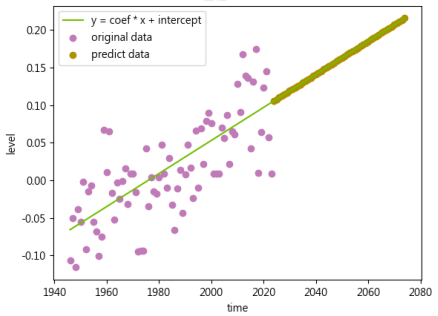
```
Month: July
Train score: 0.5731968420785034
Test score: 0.4634548586502648
Train MSE: 0.0016420788501766172
Test MSE: 0.0022613365663611273
Coefficient: 0.002125146972721544
Intercept: -4.089592141707543
----------------
Month: August
Train score: 0.5704369750426426
Test score: 0.46905540051871786
Train MSE: 0.0016946171935395022
Test MSE: 0.0029974456623692363
Coefficient: 0.002146743105800648
Intercept: -4.094637536899676
----------------
Month: September
Train score: 0.6054031103462003
Test score: 0.500776849517659
Train MSE: 0.001673626787420899
Test MSE: 0.0016914416497479635
Coefficient: 0.0022931300674622775
Intercept: -4.437350391492179
```

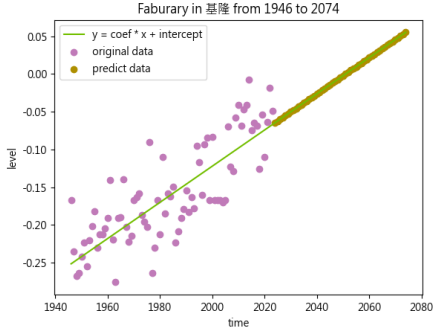
```
Month: October
Train score: 0.5341241550944142
Test score: 0.4180071897051405
Train MSE: 0.002058873244827731
Test MSE: 0.0022345744672312507
Coefficient: 0.0021986429230137308
Intercept: -4.344391707941562
----------------
Month: November
Train score: 0.4685534659508257
Test score: 0.20196956147685285
Train MSE: 0.0025331258026862658
Test MSE: 0.0028665253351751446
Coefficient: 0.0021386122578579457
Intercept: -4.319659225198702
----------------
Month: December
Train score: 0.5290430795994161
Test score: 0.5566620441533068
Train MSE: 0.0019112864364661047
Test MSE: 0.0025583816850217427
Coefficient: 0.0020968709394373823
Intercept: -4.310521123316439
```

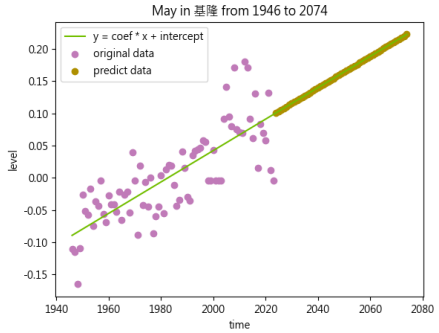

January in 基隆 from 1946 to 2074
y = coef * x + intercept
original data
predict data
level
time

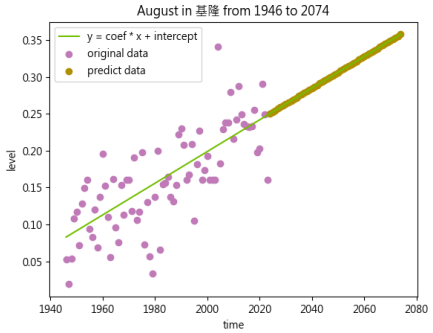

April in 基隆 from 1946 to 2074
y = coef * x + intercept
original data
predict data
level
time

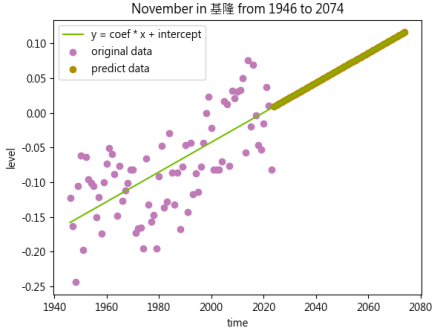

July in 基隆 from 1946 to 2074
y = coef * x + intercept
original data
predict data
level
time

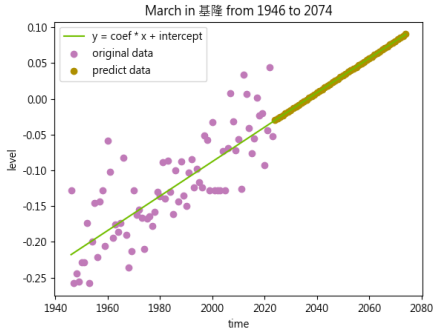

October in 基隆 from 1946 to 2074
y = coef * x + intercept
original data
predict data
level
time

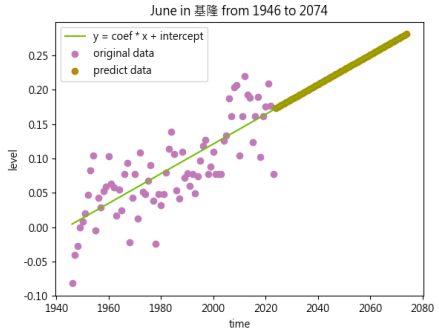

Faburary in 基隆 from 1946 to 2074
y = coef * x + intercept
original data
predict data
level
time

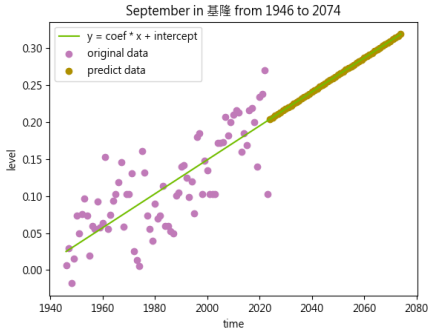

May in 基隆 from 1946 to 2074
y = coef * x + intercept
original data
predict data
level
time

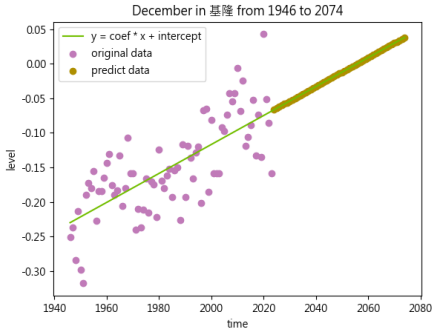

August in 基隆 from 1946 to 2074
y = coef * x + intercept
original data
predict data
level
time

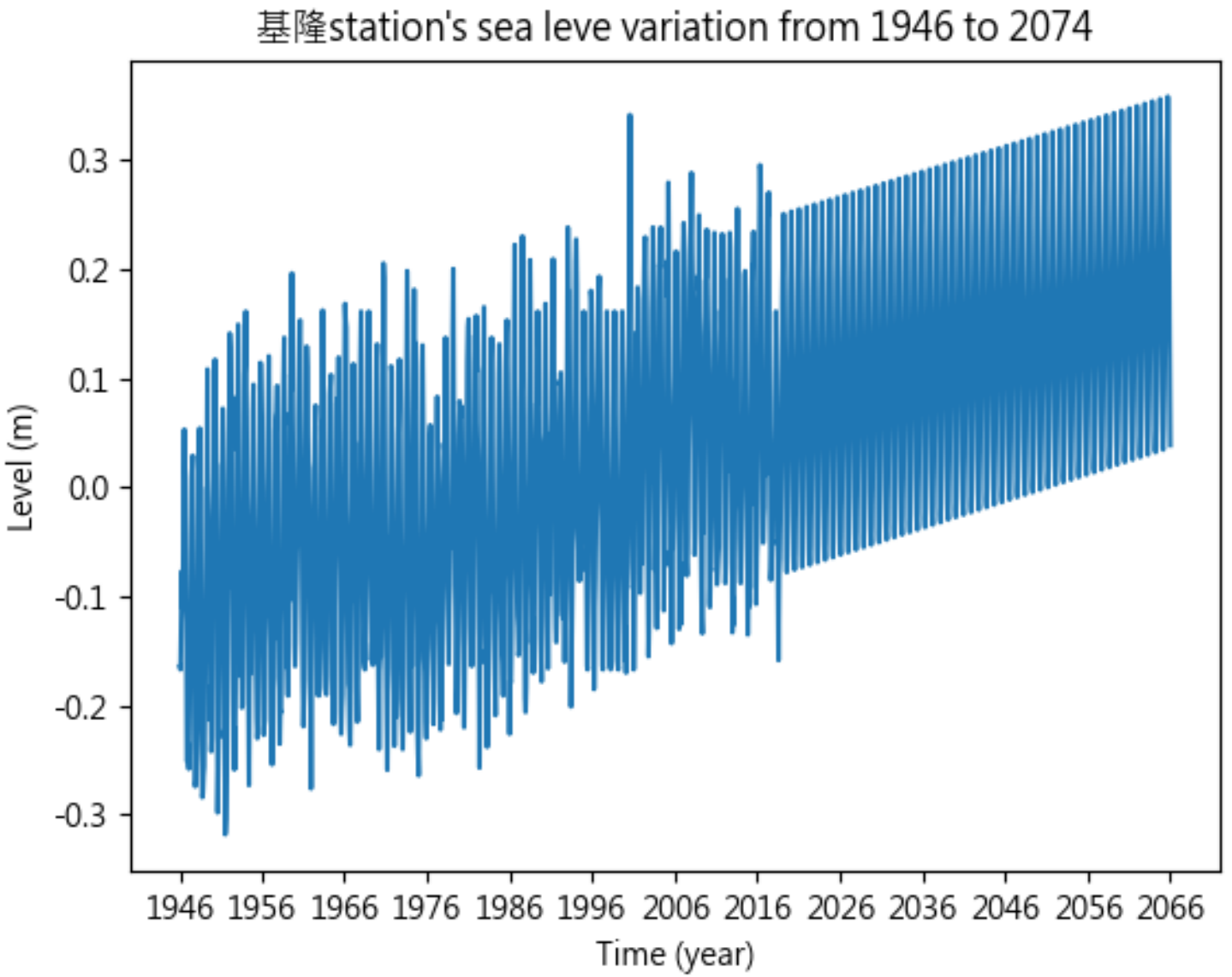

November in 基隆 from 1946 to 2074
y = coef * x + intercept
original data
predict data
level
time


March in 基隆 from 1946 to 2074
y = coef * x + intercept
original data
predict data
level
time


June in 基隆 from 1946 to 2074
y = coef * x + intercept
original data
predict data
level
time


September in 基隆 from 1946 to 2074
y = coef * x + intercept
original data
predict data
level
time


December in 基隆 from 1946 to 2074
y = coef * x + intercept
original data
predict data
level
time


基隆station's sea leve variation from 1946 to 2074
Level (m)
0.3
0.2
0.1
0.0
-0.1
-0.2
-0.3
1946 1956 1966 1976 1986 1996 2006 2016 2026 2036 2046 2056 2066
Time (year)

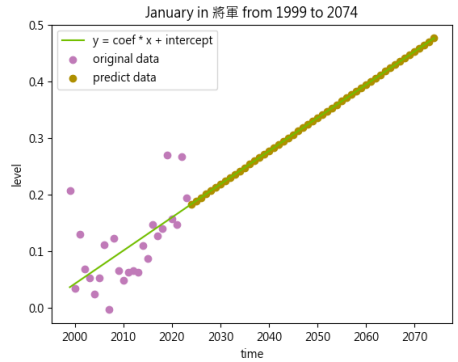
將軍 STATION

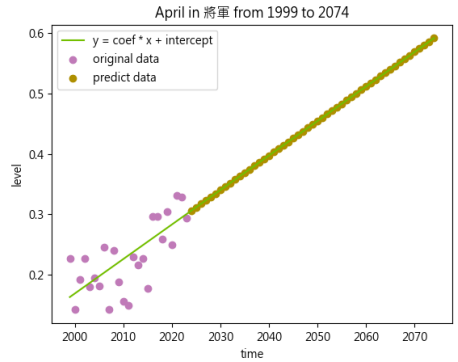
```
Month: January
Train score: 0.5018032491237838
Test score: 0.006537069076602564
Train MSE: 0.0017127456278698346
Test MSE: 0.010077529017217996
Coefficient: 0.005868636454382111
Intercept: -11.695514773407863
----------------
Month: Faburary
Train score: 0.3665310449687865
Test score: 0.3777512071613819
Train MSE: 0.001962144749450982
Test MSE: 0.002859208313141736
Coefficient: 0.004760830505090835
Intercept: -9.43870622878818
----------------
Month: March
Train score: 0.38209756273944506
Test score: 0.4364105021267558
Train MSE: 0.002140500549011779
Test MSE: 0.0021764473793069797
Coefficient: 0.005140547015372328
Intercept: -10.167554102615288
```

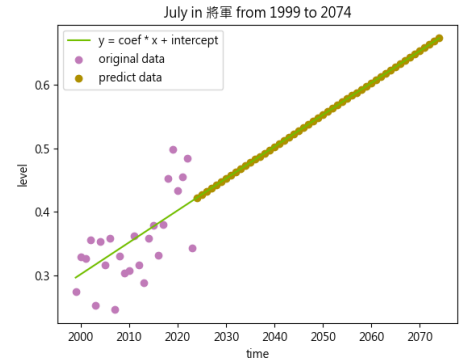
```
Month: April
Train score: 0.6067761500354611
Test score: 0.10710150247329675
Train MSE: 0.0010617781243761238
Test MSE: 0.004098261239887964
Coefficient: 0.0057192054302255945
Intercept: -11.269644040726693
----------------
Month: May
Train score: 0.4148338126955773
Test score: 0.27705454945451435
Train MSE: 0.001914378130614893
Test MSE: 0.0015345529311458698
Coefficient: 0.005205180674785386
Intercept: -10.19791385506089
----------------
Month: June
Train score: 0.4779784892486485
Test score: 0.529689143144743
Train MSE: 0.0016928097237472618
Test MSE: 0.0025858443407273
Coefficient: 0.005562737073268115
Intercept: -10.879095528049506
```

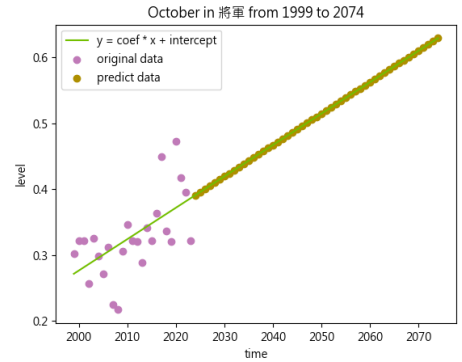
```
Month: July
Train score: 0.3614984366966183
Test score: 0.5631327488152547
Train MSE: 0.002234913500698754
Test MSE: 0.003216286601982285
Coefficient: 0.005026053104412056
Intercept: -9.750445398283086
----------------
Month: August
Train score: 0.47760859225033303
Test score: 0.7366048808769255
Train MSE: 0.0012224207077260886
Test MSE: 0.001242908888117964
Coefficient: 0.004723597524455977
Intercept: -9.119176981433416
----------------
Month: September
Train score: 0.45560872293584287
Test score: 0.7535398601429497
Train MSE: 0.0016676596576162927
Test MSE: 0.001249907811676639
Coefficient: 0.005278598522659213
Intercept: -10.247339488919943
```

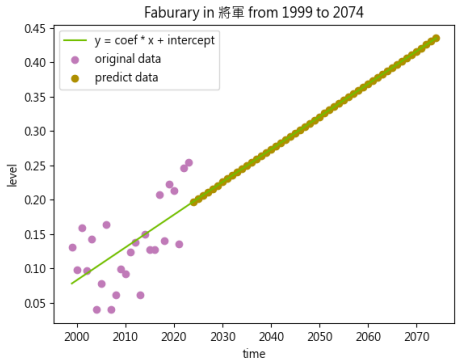
```
Month: October
Train score: 0.32839905121028334
Test score: 0.39559130826922284
Train MSE: 0.002330040158963846
Test MSE: 0.001889865097303794
Coefficient: 0.004769265322419641
Intercept: -9.262144489918139
----------------
Month: November
Train score: 0.296118343033153
Test score: -0.6213692044126147
Train MSE: 0.002246783210221606
Test MSE: 0.0011287323853438863
Coefficient: 0.004343980834497904
Intercept: -8.494079856258733
----------------
Month: December
Train score: 0.3921486026389709
Test score: 0.8338411228098116
Train MSE: 0.0032760515771611083
Test MSE: 0.0006446565653674054
Coefficient: 0.006495707726093032
Intercept: -12.918717807945695
```

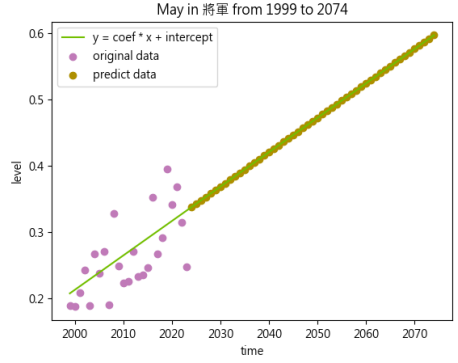

January in 將軍 from 1999 to 2074
y = coef * x + intercept
original data
predict data
level
time

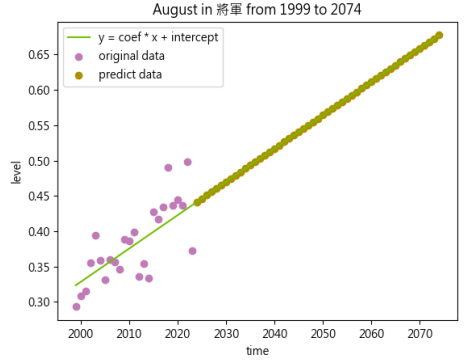

April in 將軍 from 1999 to 2074
y = coef * x + intercept
original data
predict data
level
time

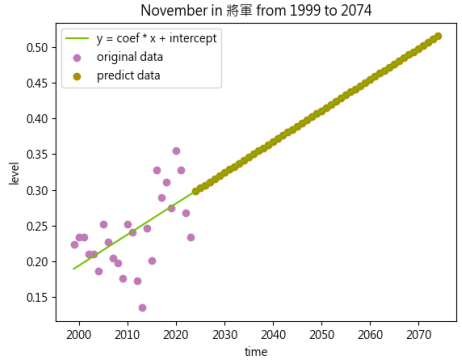

July in 將軍 from 1999 to 2074
y = coef * x + intercept
original data
predict data
level
time

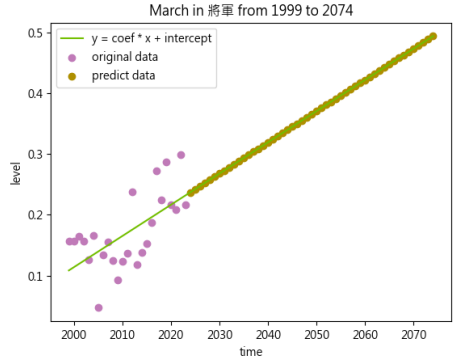

October in 將軍 from 1999 to 2074
y = coef * x + intercept
original data
predict data
level
time

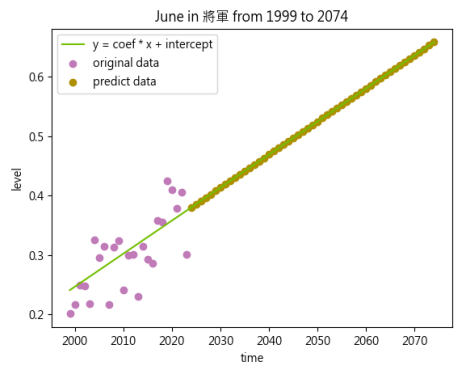

Faburary in 將軍 from 1999 to 2074
y = coef * x + intercept
original data
predict data
level
time

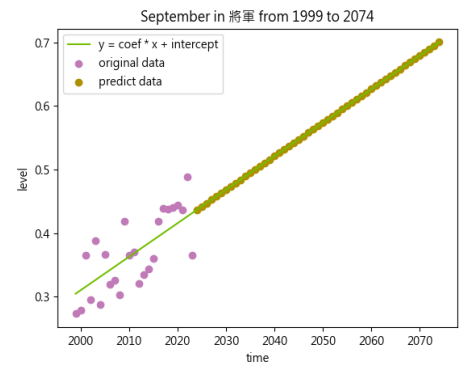

May in 將軍 from 1999 to 2074
y = coef * x + intercept
original data
predict data
level
time

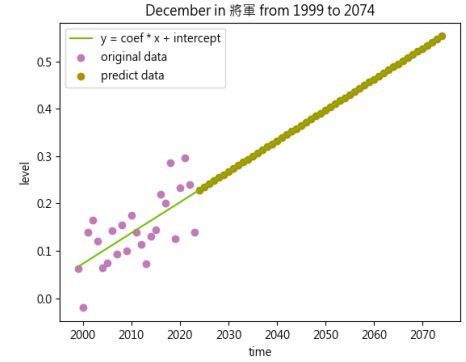

August in 將軍 from 1999 to 2074
y = coef * x + intercept
original data
predict data
level
time

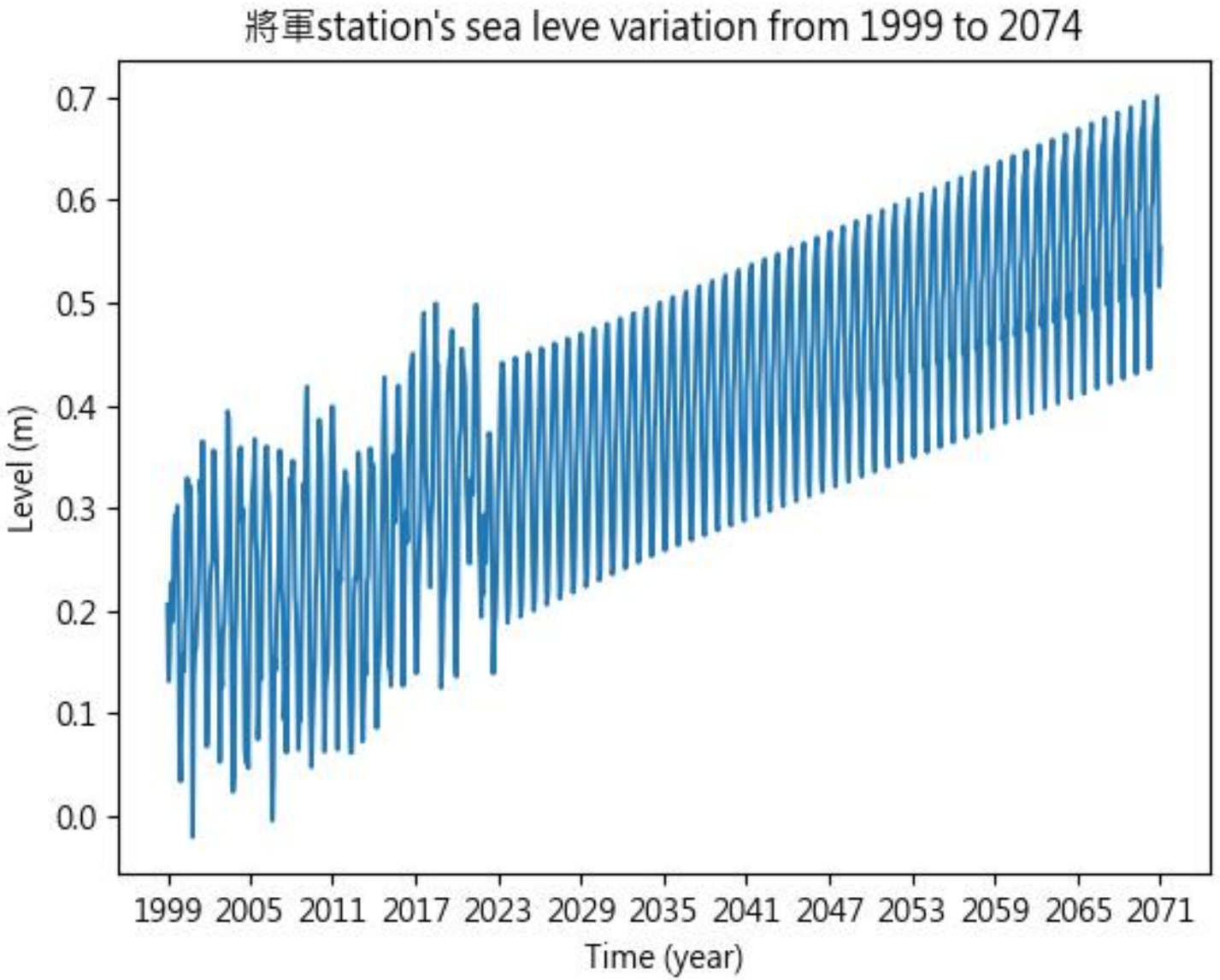

November in 將軍 from 1999 to 2074
y = coef * x + intercept
original data
predict data
level
time


March in 將軍 from 1999 to 2074
y = coef * x + intercept
original data
predict data
level
time


June in 將軍 from 1999 to 2074
y = coef * x + intercept
original data
predict data
level
time


September in 將軍 from 1999 to 2074
y = coef * x + intercept
original data
predict data
level
time


December in 將軍 from 1999 to 2074
y = coef * x + intercept
original data
predict data
level
time


將軍station's sea leve variation from 1999 to 2074
Level (m)
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
1999
2005
2011
2017
2023
2029
2035
2041
2047
2053
2059
2065
2071
Time (year)

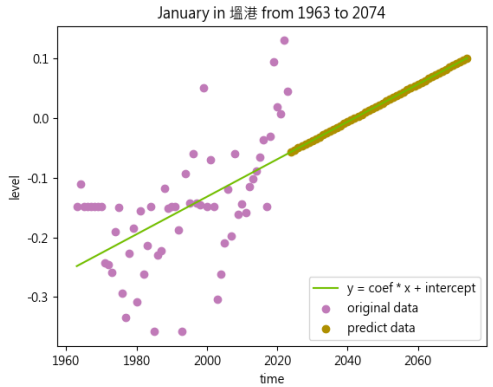
塭港 STATION

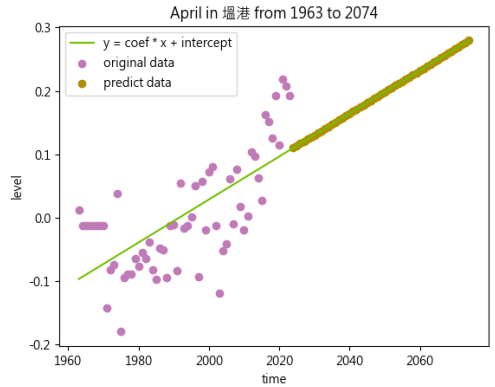
```
Month: January
Train score: 0.3055935697204553 7
Test score: 0.2176118674158613 6
Train MSE: 0.006711536929774856
Test MSE: 0.008788005721932762
Coefficient: 0.003141352248454555
Intercept: -6.414612446654401
----------------
Month: Faburary
Train score: 0.3873843972777639 7
Test score: -0.016188522178897102
Train MSE: 0.004213333737367802
Test MSE: 0.009074070441171051
Coefficient: 0.002983529749553007
Intercept: -6.074384153583129
----------------
Month: March
Train score: 0.3629377101711712 7
Test score: 0.1317744851312517 8
Train MSE: 0.004275739781136211
Test MSE: 0.009642250559249131
Coefficient: 0.002852798767993944
Intercept: -5.749574644688931
```

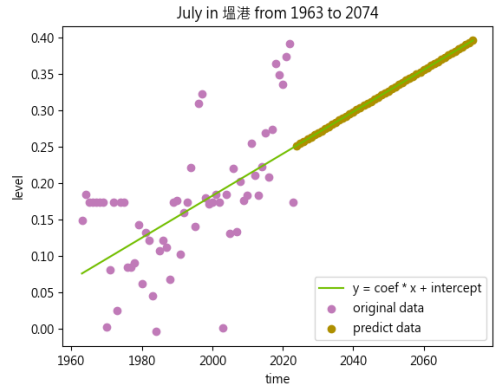
```
Month: April
Train score: 0.4960485683461844
Test score: 0.3743416246450245
Train MSE: 0.0034982707289713567
Test MSE: 0.006776805735235441
Coefficient: 0.0033918362197599508
Intercept: -6.755092596217847
----------------
Month: May
Train score: 0.3160902383423499 5
Test score: 0.3522162294046719
Train MSE: 0.005009028041786847
Test MSE: 0.006227310207455584
Coefficient: 0.0027811427378202367
Intercept: -5.485591571721284
----------------
Month: June
Train score: 0.3012663721732144
Test score: 0.2375688068082142 4
Train MSE: 0.006796421568144963
Test MSE: 0.006276367418058 62
Coefficient: 0.003128960407884491
Intercept: -6.133093366221631
```

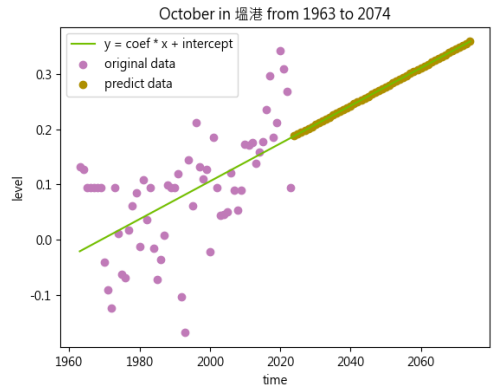
```
Month: July
Train score: 0.4035900473421465 4
Test score: 0.0890861435130813
Train MSE: 0.003678332497168672
Test MSE: 0.010580000331977408
Coefficient: 0.0028837903327547437
Intercept: -5.5849222296177405
----------------
Month: August
Train score: 0.3962707151286980 5
Test score: 0.2881678090783673 7
Train MSE: 0.004753931043514643
Test MSE: 0.005970941081996218
Coefficient: 0.0032288042319756037
Intercept: -6.243930034062879
----------------
Month: September
Train score: 0.2849687066985539
Test score: 0.2769578648086587
Train MSE: 0.006064716671959541
Test MSE: 0.006286252395672579
Coefficient: 0.00284172289434417
Intercept: -5.511876120747034
```

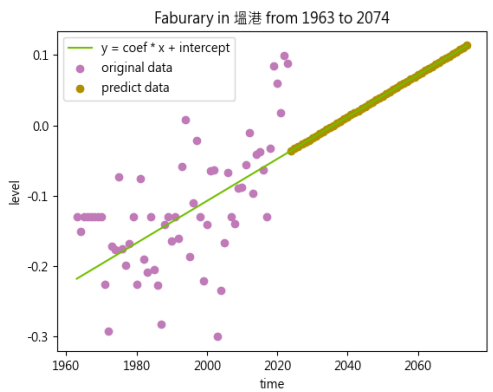
```
Month: October
Train score: 0.3152286186545417 7
Test score: 0.2484031422299729 5
Train MSE: 0.007595134345948128
Test MSE: 0.00665643529544457
Coefficient: 0.003417811884160109
Intercept: -6.729964638555004
----------------
Month: November
Train score: 0.2714718040282659
Test score: -0.1013160748951291 7
Train MSE: 0.007122315435412856 5
Test MSE: 0.009418702705703415
Coefficient: 0.002977763421775012
Intercept: -5.945673472717071
----------------
Month: December
Train score: 0.3185517059611142 6
Test score: 0.2515929941842810 5
Train MSE: 0.004922465247019777
Test MSE: 0.007281204707365528
Coefficient: 0.002772715111650549
Intercept: -5.624321881158399
```

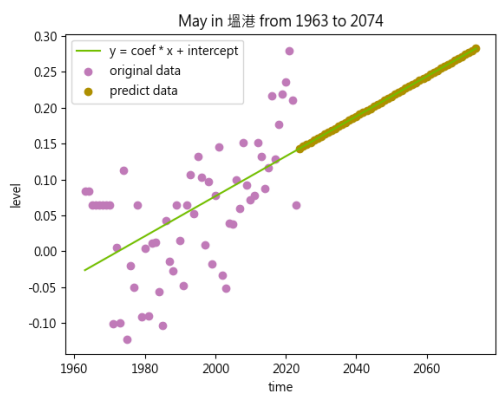

January in 堰港 from 1963 to 2074
y = coef * x + intercept
original data
predict data
level
time

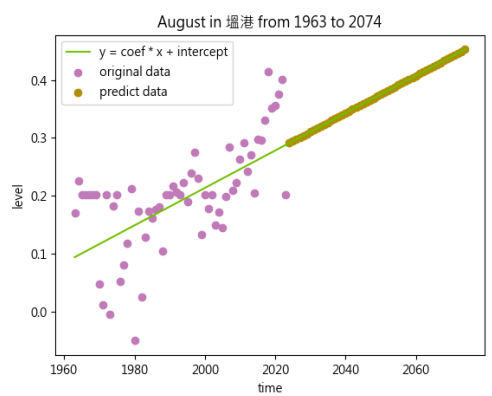

April in 堰港 from 1963 to 2074
y = coef * x + intercept
original data
predict data
level
time

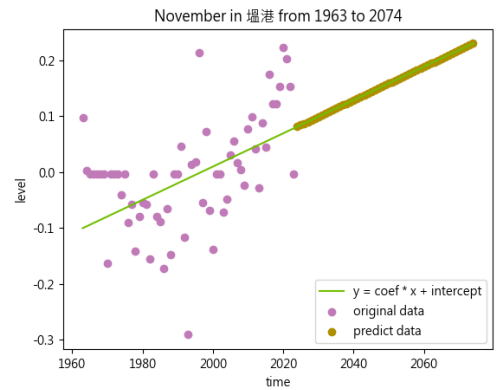

July in 堰港 from 1963 to 2074
y = coef * x + intercept
original data
predict data
level
time

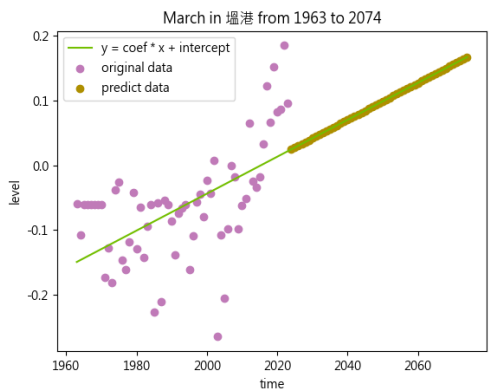

October in 堰港 from 1963 to 2074
y = coef * x + intercept
original data
predict data
level
time

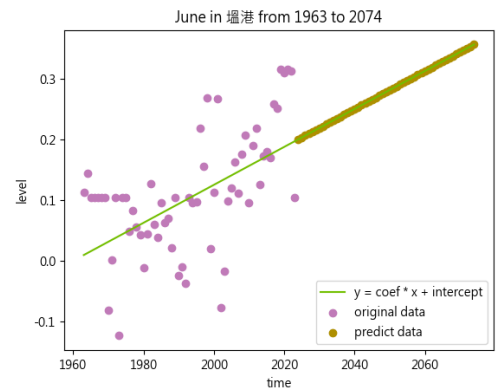

Faburary in 堰港 from 1963 to 2074
y = coef * x + intercept
original data
predict data
level
time

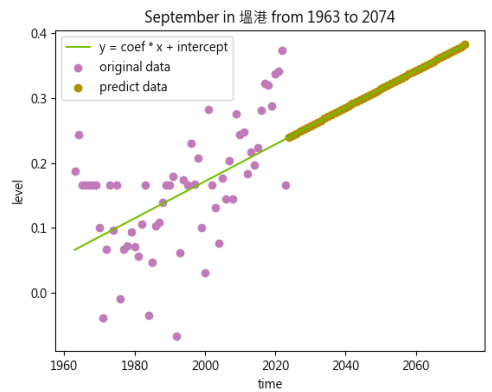

May in 堰港 from 1963 to 2074
y = coef * x + intercept
original data
predict data
level
time

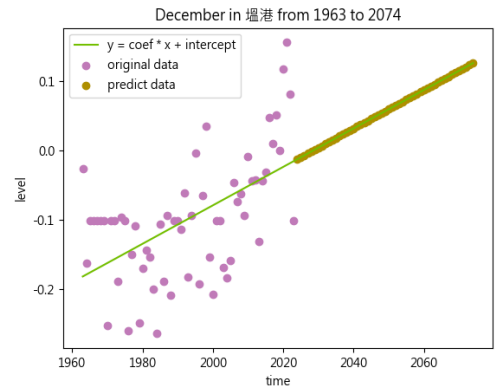

August in 堰港 from 1963 to 2074
y = coef * x + intercept
original data
predict data
level
time

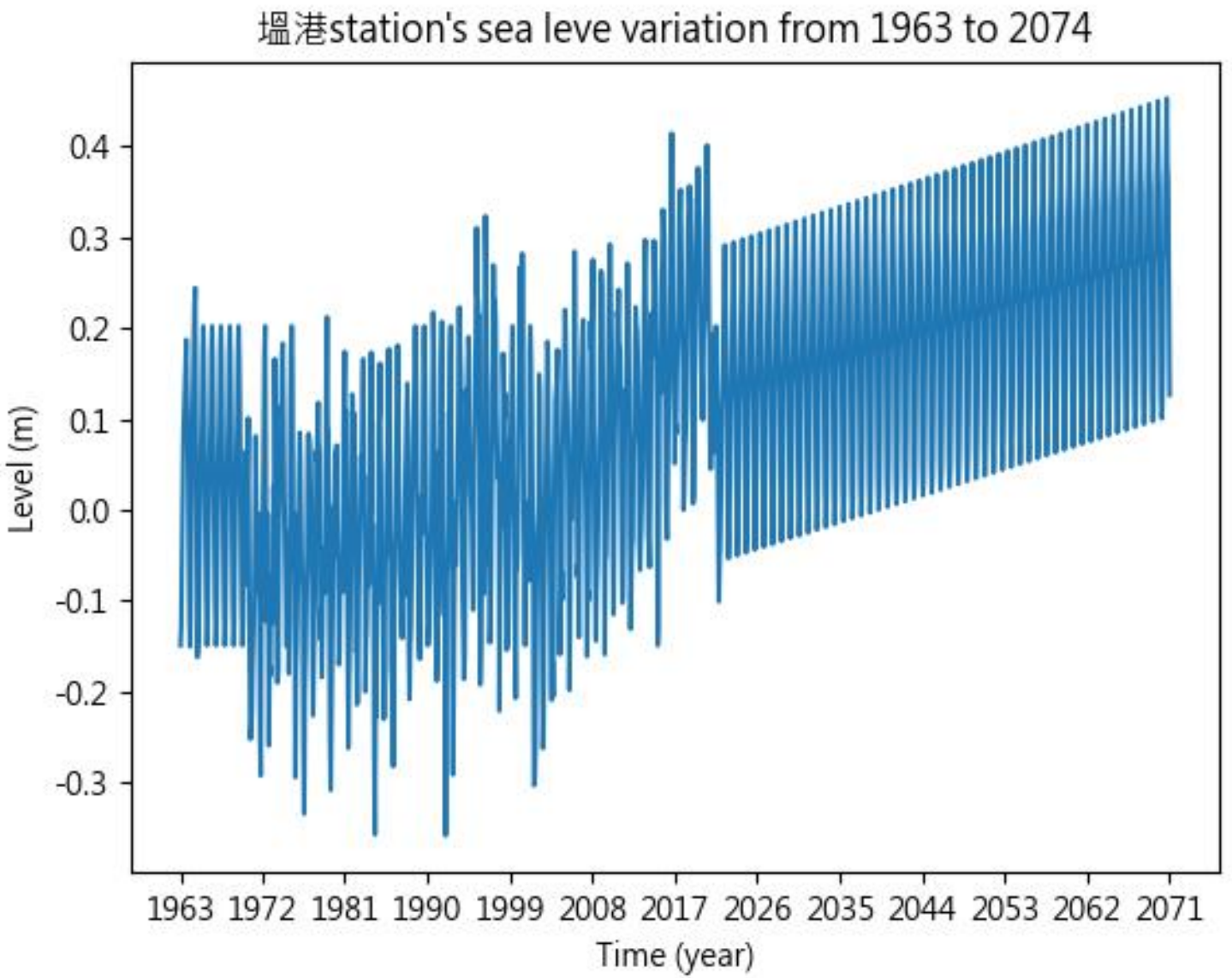

November in 堰港 from 1963 to 2074
y = coef * x + intercept
original data
predict data
level
time


March in 堰港 from 1963 to 2074
y = coef * x + intercept
original data
predict data
level
time


June in 堰港 from 1963 to 2074
y = coef * x + intercept
original data
predict data
level
time


September in 堰港 from 1963 to 2074
y = coef * x + intercept
original data
predict data
level
time


December in 堰港 from 1963 to 2074
y = coef * x + intercept
original data
predict data
level
time


堀港station's sea leve variation from 1963 to 2074
Level (m)
0.4
0.3
0.2
0.1
0.0
-0.1
-0.2
-0.3
1963 1972 1981 1990 1999 2008 2017 2026 2035 2044 2053 2062 2071
Time (year)

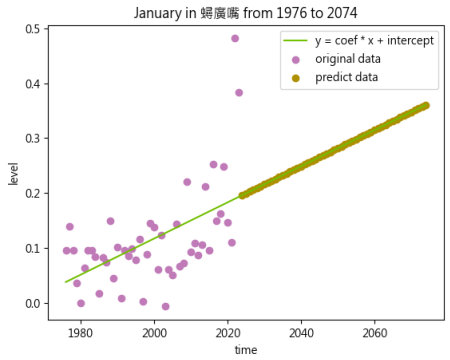
蟳廣嘴 STATION

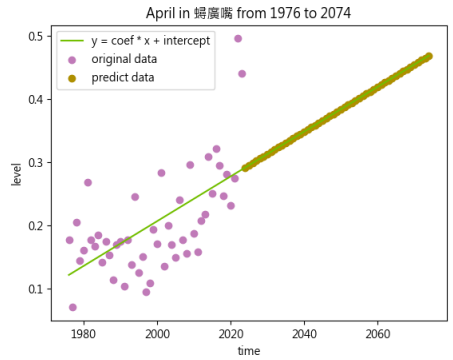
```
Month: January
Train score: 0.2817004141230046
Test score: 0.36814343927872084
Train MSE: 0.0057028803670214285
Test MSE: 0.004475591665163393
Coefficient: 0.0032986263893615073
Intercept: -6.4804305765660715
----------------
Month: Faburary
Train score: 0.243588512843265
Test score: 0.2679644303691977
Train MSE: 0.00643298962703464
Test MSE: 0.0027808860033362768
Coefficient: 0.0031746864144955164
Intercept: -6.2065396147017085
----------------
Month: March
Train score: 0.2317430997010439
Test score: 0.36355357155252277
Train MSE: 0.005701586131502969
Test MSE: 0.0022275640906822417
Coefficient: 0.0028926312881347926
Intercept: -5.612588947774492
```

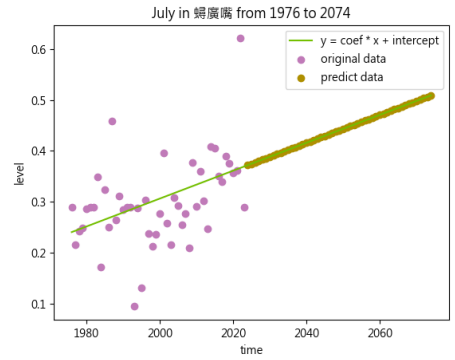
```
Month: April
Train score: 0.3646329548972511
Test score: 0.4796540864259129
Train MSE: 0.004503863539923152
Test MSE: 0.0025269506566307034
Coefficient: 0.0035461188845351717
Intercept: -6.8857801144843975
----------------
Month: May
Train score: 0.4076776685955168
Test score: 0.18724162842700576
Train MSE: 0.0024798402999199817
Test MSE: 0.0033600162563362
Coefficient: 0.0028816174711683257
Intercept: -5.545915302433548
----------------
Month: June
Train score: 0.27439019312288704
Test score: -0.0931538353273631
Train MSE: 0.004145888839686191
Test MSE: 0.004311890645757018
Coefficient: 0.0027617642996000794
Intercept: -5.264719853508485
```

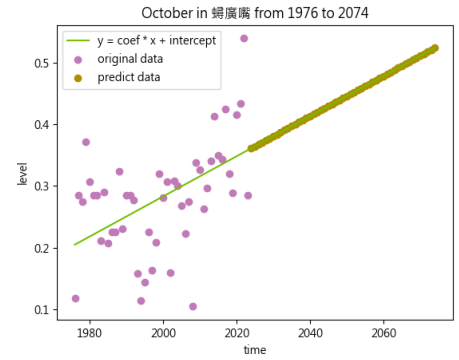
```
Month: July
Train score: 0.20850652963190153
Test score: -0.0374839056659122
Train MSE: 0.0058686052106697644
Test MSE: 0.005777498874516104
Coefficient: 0.002742512861806751
Intercept: -5.1790272228941845
----------------
Month: August
Train score: 0.2855074388158305
Test score: 0.2202458193052378
Train MSE: 0.005263645421803243
Test MSE: 0.003616000847386663
Coefficient: 0.003198882109369997
Intercept: -6.070075951200938
----------------
Month: September
Train score: 0.20442544275881525
Test score: 0.1279710605122102
Train MSE: 0.00563980820201402
Test MSE: 0.007021018922234409
Coefficient: 0.002655242934008065
Intercept: -4.990463105903121
```

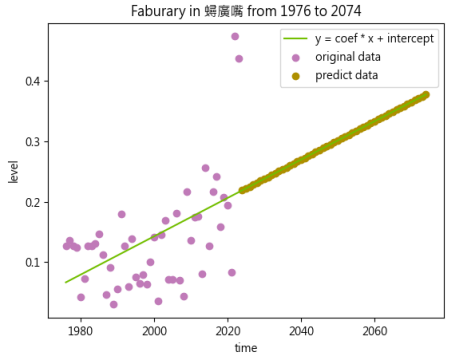
```
Month: October
Train score: 0.2634087239643267
Test score: -0.12311973303849588
Train MSE: 0.0060703636069075785
Test MSE: 0.0050230519252387005
Coefficient: 0.003249781846237815
Intercept: -6.21646524364153
----------------
Month: November
Train score: 0.2977199586515875
Test score: -0.13134870337338578
Train MSE: 0.005557717433158774
Test MSE: 0.0064480653674414455
Coefficient: 0.003385650367744912
Intercept: -6.569476744303573
----------------
Month: December
Train score: 0.21277990182259354
Test score: 0.05088757129556476
Train MSE: 0.009397496074118905
Test MSE: 0.00390826465004341
Coefficient: 0.0035153482542645263
Intercept: -6.883719422254714
```

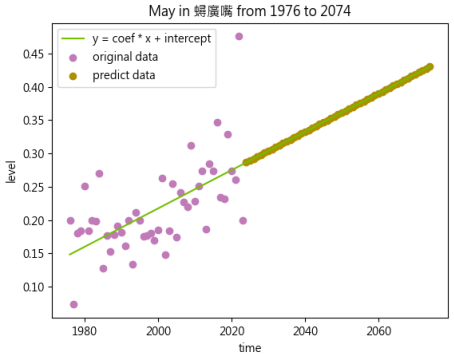

January in 蝠廣嘴 from 1976 to 2074
y = coef * x + intercept
original data
predict data
level
time

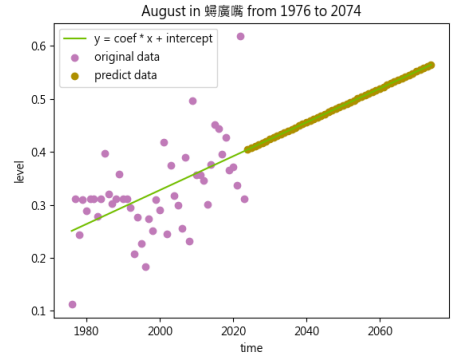

April in 蝠廣嘴 from 1976 to 2074
y = coef * x + intercept
original data
predict data
level
time

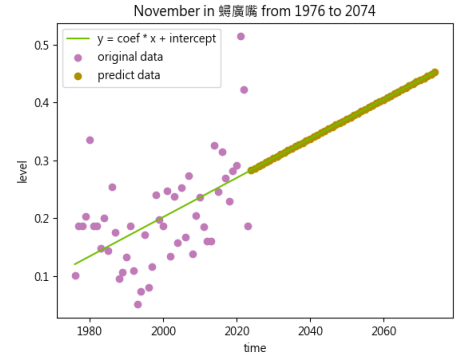

July in 蝠廣嘴 from 1976 to 2074
y = coef * x + intercept
original data
predict data
level
time

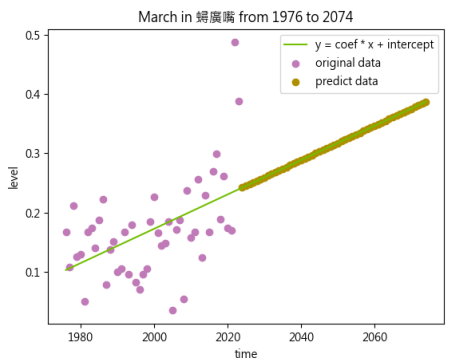

October in 蝠廣嘴 from 1976 to 2074
y = coef * x + intercept
original data
predict data
level
time

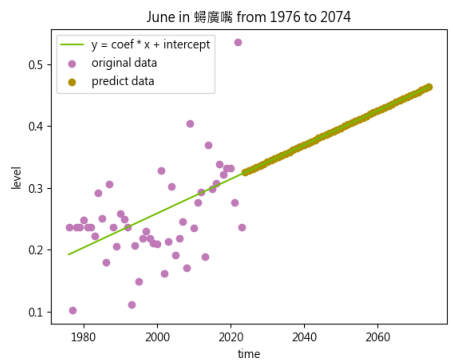

Faburary in 蝠廣嘴 from 1976 to 2074
y = coef * x + intercept
original data
predict data
level
time

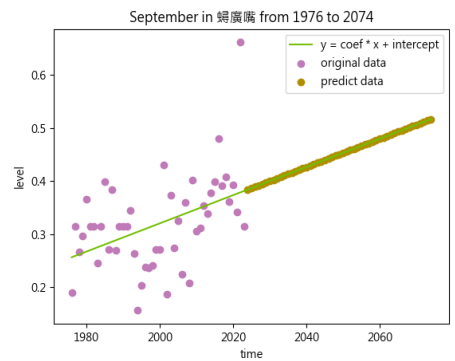

May in 蝠廣嘴 from 1976 to 2074
y = coef * x + intercept
original data
predict data
level
time

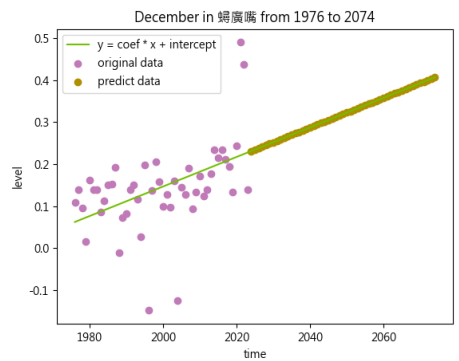

August in 蝠廣嘴 from 1976 to 2074
y = coef * x + intercept
original data
predict data
level
time

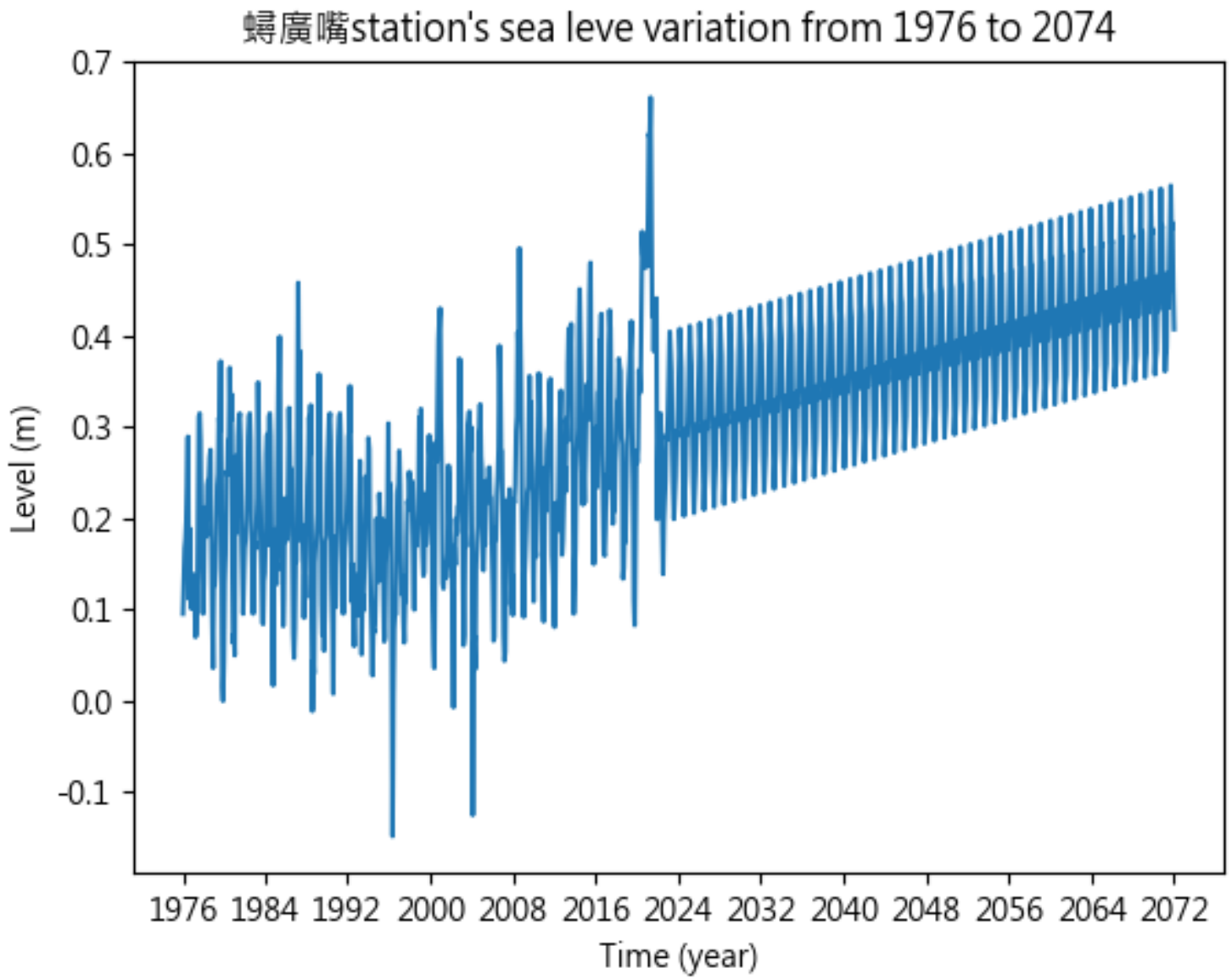

November in 蝠廣嘴 from 1976 to 2074
y = coef * x + intercept
original data
predict data
level
time


March in 蝠廣嘴 from 1976 to 2074
y = coef * x + intercept
original data
predict data
level
time


June in 蝠廣嘴 from 1976 to 2074
y = coef * x + intercept
original data
predict data
level
time


September in 蝠廣嘴 from 1976 to 2074
y = coef * x + intercept
original data
predict data
level
time


December in 蝠廣嘴 from 1976 to 2074
y = coef * x + intercept
original data
predict data
level
time


鱟廣嘴station's sea leve variation from 1976 to 2074
Level (m)
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
-0.1
1976 1984 1992 2000 2008 2016 2024 2032 2040 2048 2056 2064 2072
Time (year)

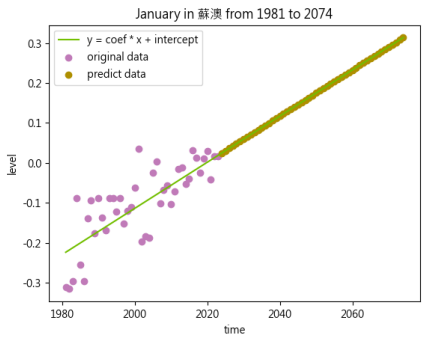
蘇澳 STATION

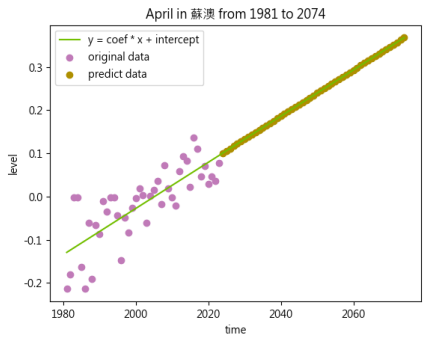
```
Month: January
Train score: 0.5571720330431564
Test score: 0.7237179557603979
Train MSE: 0.003954554875186962
Test MSE: 0.0022321747294264923
Coefficient: 0.005783572541813023
Intercept: -11.681256135290425
----------------
Month: Faburary
Train score: 0.6517586380912277
Test score: 0.4922736086044278
Train MSE: 0.002550877917127902
Test MSE: 0.0019144669798222373
Coefficient: 0.005665239945102237
Intercept: -11.422841040731353
----------------
Month: March
Train score: 0.5717065775622596
Test score: 0.7324132908509878
Train MSE: 0.002984030311447667
Test MSE: 0.0017279560402580318
Coefficient: 0.005174730745074322
Intercept: -10.413333248040198
```

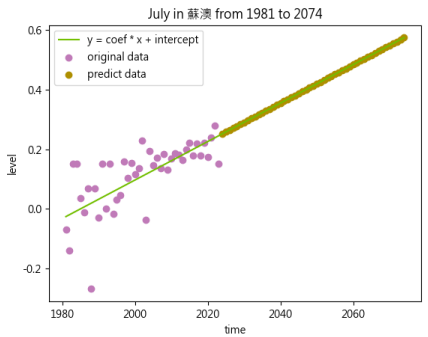
```
Month: April
Train score: 0.5724833474998046
Test score: 0.691021133026277
Train MSE: 0.003177120698454993
Test MSE: 0.0016595674546089124
Coefficient: 0.0053480076298588015
Intercept: -10.723668915303914
----------------
Month: May
Train score: 0.5351802210425354
Test score: 0.6233761213413977
Train MSE: 0.004649036958154822
Test MSE: 0.0015285442854072375
Coefficient: 0.00599873223382726
Intercept: -11.970610109562916
----------------
Month: June
Train score: 0.4934179713688849
Test score: -0.20644615584704185
Train MSE: 0.006902722217927158
Test MSE: 0.003112899181231112
Coefficient: 0.006722998906697063
Intercept: -13.377952929819259
```

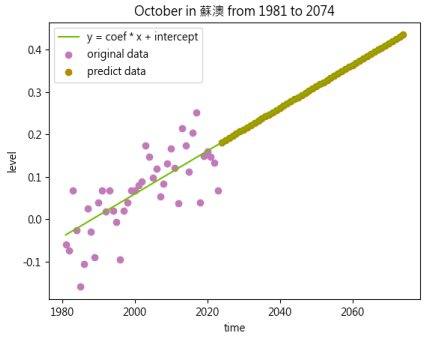
```
Month: July
Train score: 0.45897891463162654
Test score: 0.3844458723076718
Train MSE: 0.007284730039609991
Test MSE: 0.0021487778688351142
Coefficient: 0.006445643071483405
Intercept: -12.794372449929057
----------------
Month: August
Train score: 0.46113979551553885
Test score: -0.7671992227237308
Train MSE: 0.004899071788456839
Test MSE: 0.0036111734339013485
Coefficient: 0.005308910418944383
Intercept: -10.467871973062879
----------------
Month: September
Train score: 0.5440194547315131
Test score: 0.630487826173582
Train MSE: 0.004821604552737843
Test MSE: 0.00259028033852319
Coefficient: 0.006218707111121449
Intercept: -12.307188071367092
```

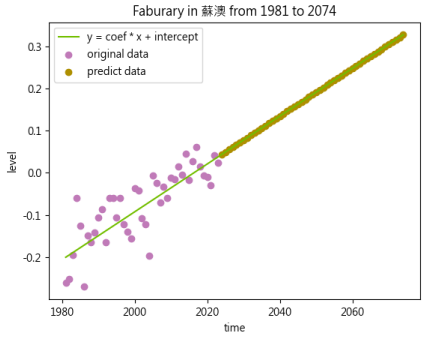
```
Month: October
Train score: 0.5622522922563504
Test score: 0.5507010008652833
Train MSE: 0.0029773842977786485
Test MSE: 0.006222702387596245
Coefficient: 0.00507039009979297
Intercept: -10.081108870873944
----------------
Month: November
Train score: 0.5581392519111985
Test score: 0.5802765300394345
Train MSE: 0.0030724519555618084
Test MSE: 0.002394828120627837
Coefficient: 0.005107888064388566
Intercept: -10.23980847426086
----------------
Month: December
Train score: 0.5201661392070065
Test score: 0.674256803134243
Train MSE: 0.003035191680703545
Test MSE: 0.0022270056989474574
Coefficient: 0.004703150806020145
Intercept: -9.495013957058784
```

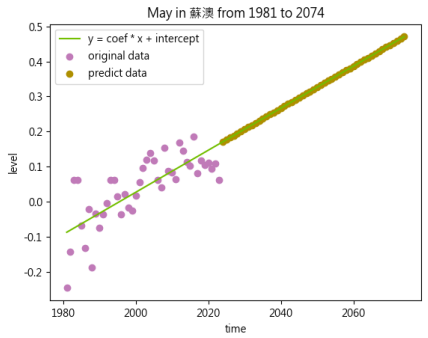

January in 蘇澳 from 1981 to 2074
y = coef * x + intercept
original data
predict data
level
time

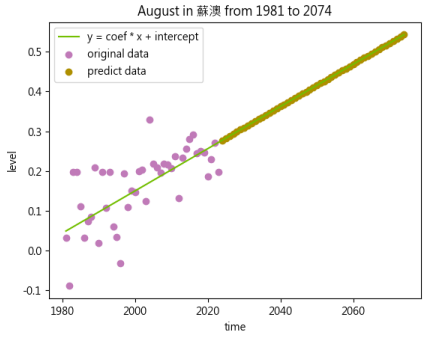

April in 蘇澳 from 1981 to 2074
y = coef * x + intercept
original data
predict data
level
time

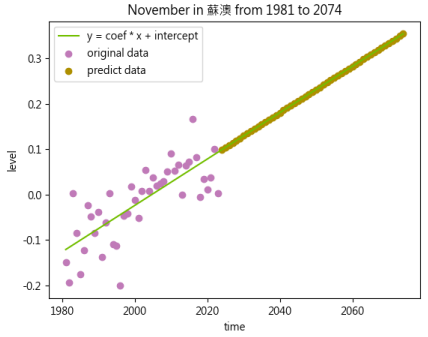

July in 蘇澳 from 1981 to 2074
y = coef * x + intercept
original data
predict data
level
time

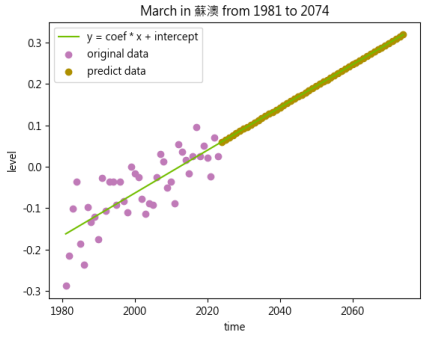

October in 蘇澳 from 1981 to 2074
y = coef * x + intercept
original data
predict data
level
time

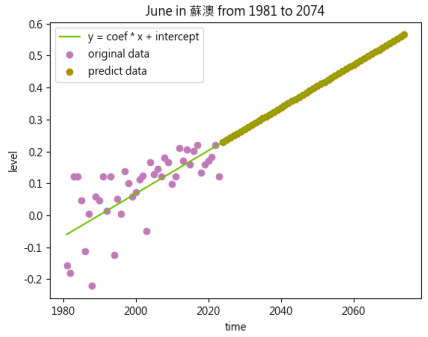

Faburary in 蘇澳 from 1981 to 2074
y = coef * x + intercept
original data
predict data
level
time

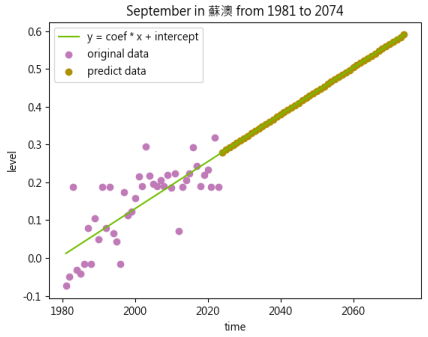

May in 蘇澳 from 1981 to 2074
y = coef * x + intercept
original data
predict data
level
time

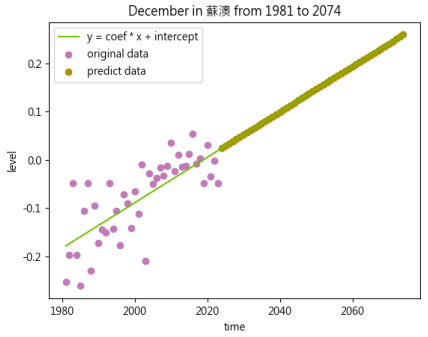

August in 蘇澳 from 1981 to 2074
y = coef * x + intercept
original data
predict data
level
time

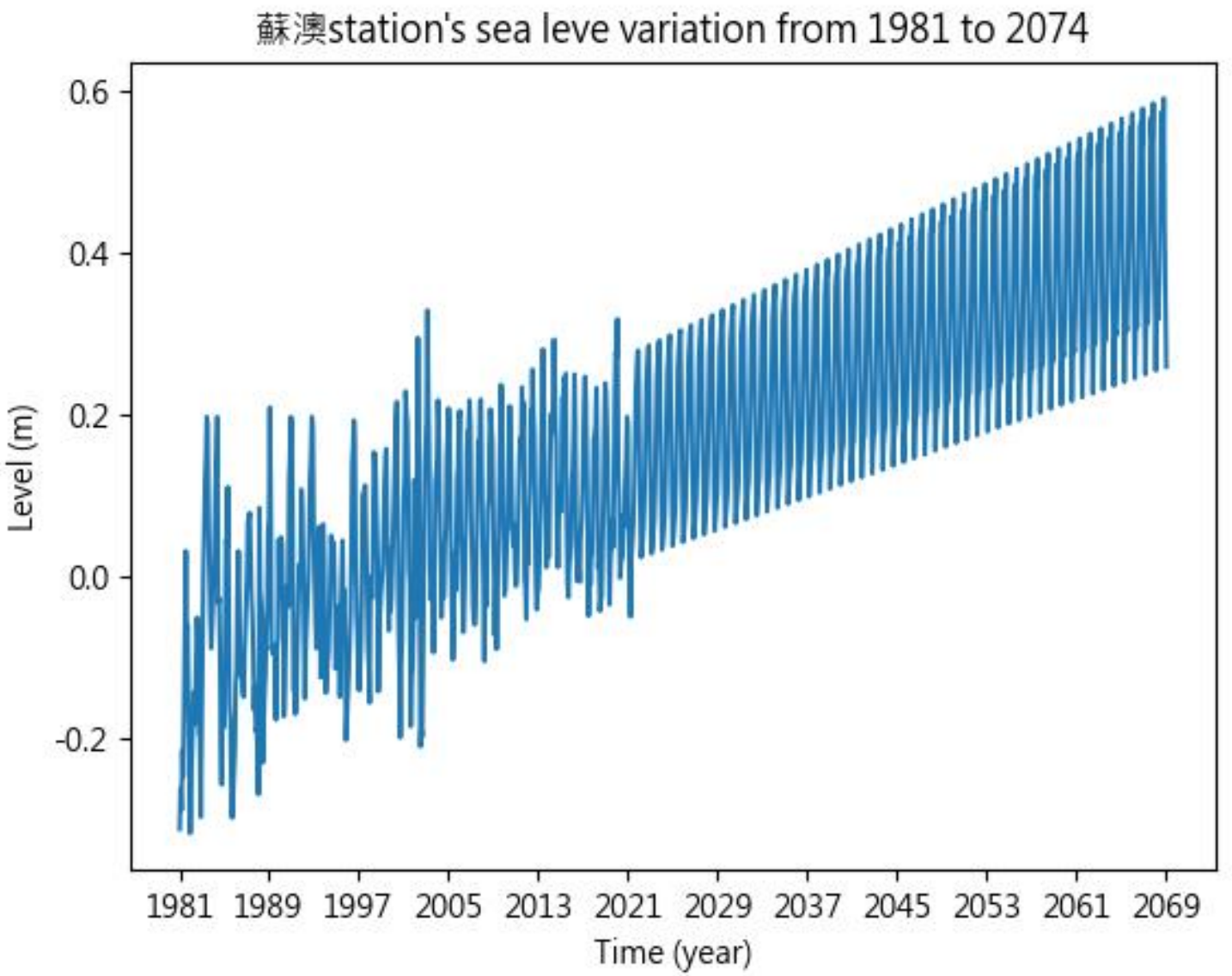

November in 蘇澳 from 1981 to 2074
y = coef * x + intercept
original data
predict data
level
time


March in 蘇澳 from 1981 to 2074
y = coef * x + intercept
original data
predict data
level
time


June in 蘇澳 from 1981 to 2074
y = coef * x + intercept
original data
predict data
level
time


September in 蘇澳 from 1981 to 2074
y = coef * x + intercept
original data
predict data
level
time


December in 蘇澳 from 1981 to 2074
y = coef * x + intercept
original data
predict data
level
time


蘇澳station's sea leve variation from 1981 to 2074
Level (m)
0.6
0.4
0.2
0.0
-0.2
1981
1989
1997
2005
2013
2021
2029
2037
2045
2053
2061
2069
Time (year)

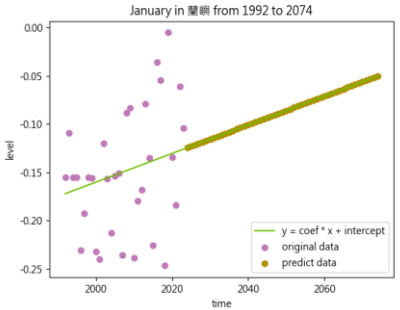
蘭嶼 STATION

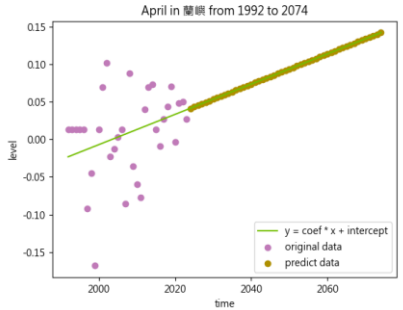
```
Month: January
Train score: 0.0586228008950521 2
Test score: 0.1630611157881432
Train MSE: 0.0030542779887485026
Test MSE: 0.005824240614844919
Coefficient: 0.0014905707984449769
Intercept: -3.1414597413277527
----------------
Month: Faburary
Train score: 0.12765963659504131
Test score: 0.19879080571301255
Train MSE: 0.005237274724880387
Test MSE: 0.003139987886034844
Coefficient: 0.0029921501196172253
Intercept: -6.13004874401914
----------------
Month: March
Train score: 0.07829637736162498
Test score: 0.3538713091613618
Train MSE: 0.008705083691499695
Test MSE: 0.0017070456285960562
Coefficient: 0.002939060630980860 7
Intercept: -5.973941499700955
```

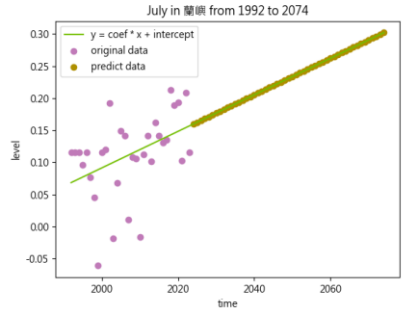
```
Month: April
Train score: 0.0993515775426439
Test score: -0.004470778088833338
Train MSE: 0.0031463842815490414
Test MSE: 0.002606458173315081
Coefficient: 0.0020135503887559808
Intercept: -4.034189668062201
----------------
Month: May
Train score: 0.05657041248777728
Test score: -1.1507542576425633
Train MSE: 0.008104020155315483
Test MSE: 0.00235268374666878
Coefficient: 0.0023825228020334928
Intercept: -4.738494953648325
----------------
Month: June
Train score: 0.10505599558111112
Test score: -0.18443600103100088
Train MSE: 0.004943160920305026
Test MSE: 0.0028709761778460004
Coefficient: 0.002603524970095694
Intercept: -5.139769063995216
```

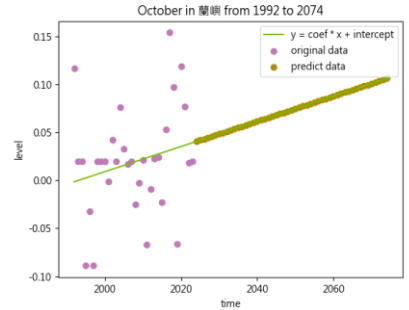
```
Month: July
Train score: 0.16190274788034953
Test score: 0.16506814783915125
Train MSE: 0.0035972126368121913
Test MSE: 0.002403104264774466
Coefficient: 0.0028491140849282296
Intercept: -5.606958358253588
----------------
Month: August
Train score: 0.005662428266767328
Test score: -0.09188050820356608
Train MSE: 0.008052861578947367
Test MSE: 0.0025267451956371176
Coefficient: -0.0007319078947368423
Intercept: 1.6052171052631583
----------------
Month: September
Train score: 0.11764051617230886
Test score: 0.11076690436104253
Train MSE: 0.0028381713978394302
Test MSE: 0.0019656134626815217
Coefficient: 0.002102431593899521
Intercept: -4.116015139055023
```

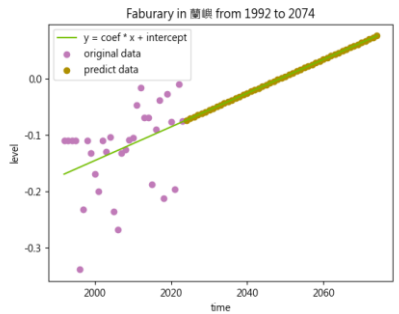
```
Month: October
Train score: 0.03977853793149122
Test score: 0.3334209675762595
Train MSE: 0.00356904715236244
Test MSE: 0.00048073271708788483
Coefficient: 0.0013142008074162675
Intercept: -2.6194227721291856
----------------
Month: November
Train score: 0.0009666720725181044
Test score: -0.1480929119908896
Train MSE: 0.004907322038726077
Test MSE: 0.01958463750209243
Coefficient: -0.00023551510167464112
Intercept: 0.41043518241626786
----------------
Month: December
Train score: 0.006960997867841212
Test score: 0.07310515137985951
Train MSE: 0.004268294538726076
Test MSE: 0.008422409746090211
Coefficient: 0.0005911894437799044
Intercept: -1.313417090311005
```

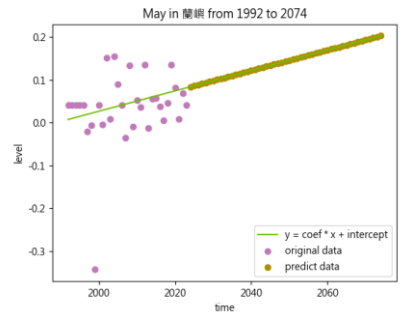

January in 蘭嶼 from 1992 to 2074
y = coef * x + intercept
original data
predict data
level
time

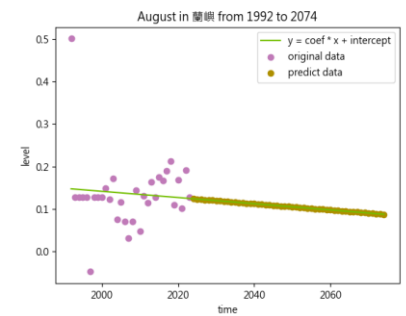

April in 蘭嶼 from 1992 to 2074
y = coef * x + intercept
original data
predict data
level
time

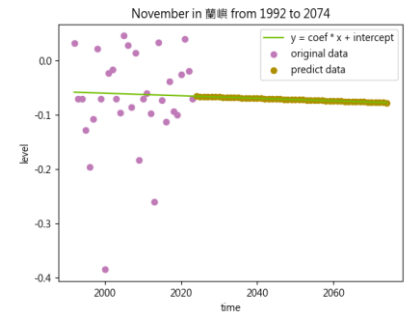

July in 蘭嶼 from 1992 to 2074
y = coef * x + intercept
original data
predict data
level
time

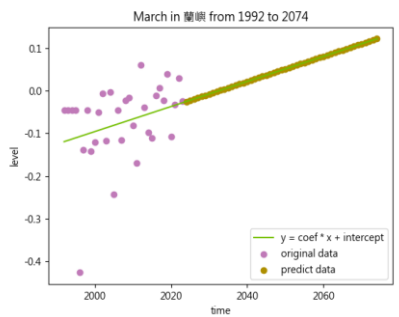

October in 蘭嶼 from 1992 to 2074
y = coef * x + intercept
original data
predict data
level
time

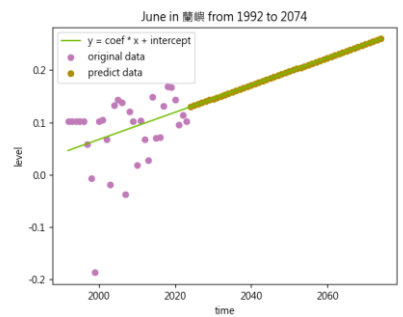

Faburary in 蘭嶼 from 1992 to 2074
y = coef * x + intercept
original data
predict data
level
time

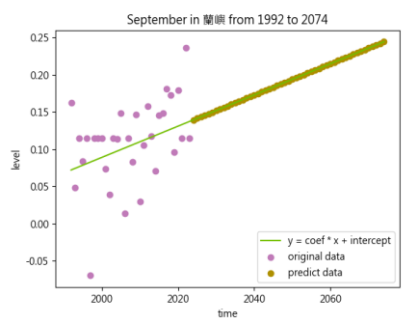

May in 蘭嶼 from 1992 to 2074
y = coef * x + intercept
original data
predict data
level
time

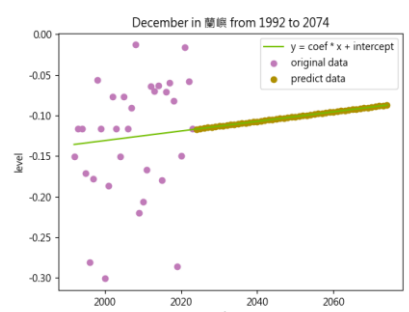

August in 蘭嶼 from 1992 to 2074
y = coef * x + intercept
original data
predict data
level
time

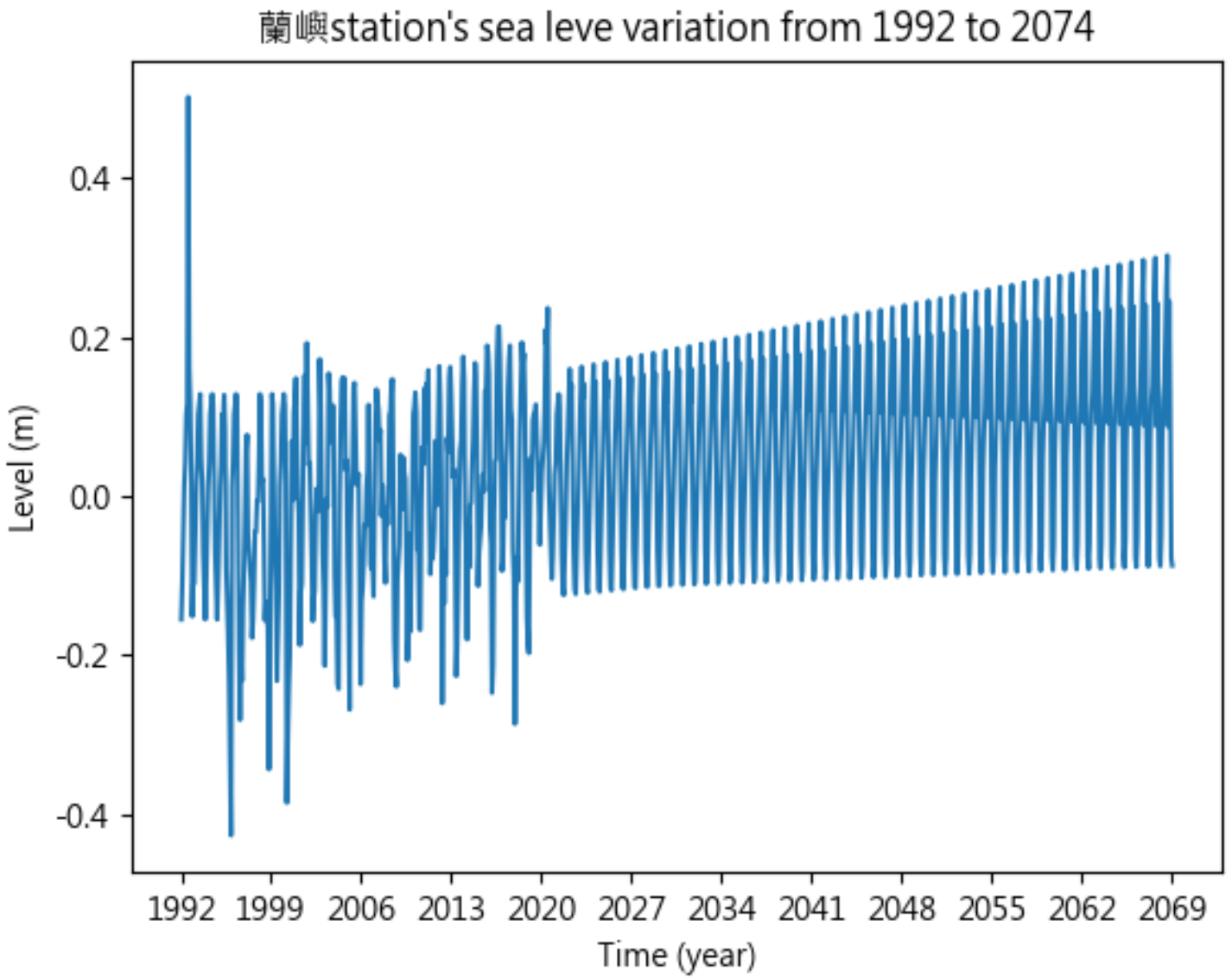

November in 蘭嶼 from 1992 to 2074
y = coef * x + intercept
original data
predict data
level
time


March in 蘭嶼 from 1992 to 2074
y = coef * x + intercept
original data
predict data
level
time


June in 蘭嶼 from 1992 to 2074
y = coef * x + intercept
original data
predict data
level
time


September in 蘭嶼 from 1992 to 2074
y = coef * x + intercept
original data
predict data
level
time


December in 蘭嶼 from 1992 to 2074
y = coef * x + intercept
original data
predict data
level
time


蘭嶼station's sea leve variation from 1992 to 2074
Level (m)
0.4
0.2
0.0
-0.2
-0.4
1992
1999
2006
2013
2020
2027
2034
2041
2048
2055
2062
2069
Time (year)

ANOVA

- f_statistic: 861.7878982793461
- p_value: 0.00e+00
- p_value is zero!

5. DISCUSSION

- The formula doesn't factor in the potential errors or uncertainties. (y = coefficient * x + intercept + error).
- Imputation methods aren't good enough to generate data that is closer to the actual values.
- The way using ANOVA is quite rudimentary.
- The original dataset contains an excessive amount of missing values.
- Maybe we should use model " statsmodels.regression.linear_model.OLS", because it support p-value.
- We can go to https://flood.firetree.net/ and see that in the coming 50 years, Taiwan is safe.

6. REFERENCE / CITED

- Data source: https://opendata.cwb.gov.tw/dataset/forecast/C-B0048-001
- Visualized website: https://flood.firetree.net/
- Python libraries:
  - https://docs.python.org/3/library/statistics.html
  - https://scikit-learn.org/stable/
  - https://scipy.org/

THANKS FOR LISTENING