WORCESTER POLYTECHNIC INSTITUTE

PORTFOLIO VALUATION AND RISK MANAGEMENT

Regression Project

Author:
Yijiang XU
Weixi LIU

Instructor:
Professor Marcel Blais

December 15, 2017

Contents

List of Figures

Chapter 1

Introduction

In this project, first we download the U.S. real GDP data and relevant parameter data from Bloomberg Terminal. Then we construct a multiple linear regression to fit our model with five parameters. Furthermore, we analysis our model using ANOVA, VIF, Cp, leverage, residual and Cook's Distance to tell how relevant is our choosing parameter to our response variable-GDP.

Chapter 2

Data Collection

Through Bloomberg Terminal, we download data from 1987-2017(30 years data) about U.S. real GDP growth rate, CPI, PPI, U.S. interest rate, unemployment rate and US Consumer Spending. All of them were calculated quarterly, and we created a program which extracted all data. Python Pandas Package helps us to read excel file into data frame.

Chapter 3

Methodology

3.1 Gross Domestic Product

3.1.1 Definition

The Gross Domestic Product[1] (GDP) is the total value of everything produced by all the people and companies in this country, which is the common chosen way to describe how a country's economy performs. When doing the calculation of GDP, there is no need to distinguish whether they are citizens or domestic-owned firms, the calculation involves all people or firms located within the country's boundaries.

3.1.2 Formula

The formula[2] for calculating the country's gross domestic product is:

$$C + I + G + (X - M),$$

in which

- C stands for Personal Consumption Expenditures,
- I means business investment,
- G indicates Government Spending,
- X-M gives out the Net Exports which is Exports minus Imports.

3.1.3 Two types of GDP

1. The first type is **Nominal GDP**. It indicates its using a raw measurement which includes the price increases. Often it chooses Bureau of Economic Analysis[3] (BEA) to measure the Nominal GDP and usually calculate it quarterly.
2. The second type is **Real GDP**, whichis used to compare the economic output from one year to another.

[1] https://www.thebalance.com/what-is-gdp-definition-of-gross-domestic-product-3306038
[2] https://www.thebalance.com/what-is-gdp-definition-of-gross-domestic-product-3306038
[3] https://www.thebalance.com/bureau-of-economic-analysis-3305976

The main difference compared to the Nominal GDP is about the inflation effects. It uses BEA measurement to calculate and contains a price deflator. Notice that usually the Real GDP is smaller than the Nominal GDP.

BEA contains three important figures:

- The income made by people or firms outside the country would not be included, which removes the exchange rates and trade policies impact out of the calculation.
- Also takes out the consideration of the inflation impact.
- Only counts the final product. For instance, a U.S. coat manufacturer uses buttons and other materials made in U.S.

Note: When doing the calculation, only the value of the coats get counted, not the buttons or other materials.

In our project we took the data form Bloomberg Terminal about US's GDP growth rate. By definition[4], it is the percent increase in GDP from quarter to quarter. It directly shows the picture about how fast a country's economy is growing.

3.2 Multiple Linear Regression

In our project, we used Multiple Linear Regression to build model for fitting our data.

The multiple linear regression (Ruppert, 2004, p. 169) model relating Y to the predictor variables is

$$Y_i = \beta_0 + \beta_1 X_{i1} + \cdots + \beta_p X_{ip} + \epsilon_i$$

where Y_i be the value of the response variable for the i-th observation, $X_1, \cdots, X_p$ be the values of predictor variables 1 through p for the i-th observation, and ϵ_i is white noise (Ruppert, 2004, p. 103).

3.2.1 Setting

Define

$$\boldsymbol{Y} = \begin{pmatrix} Y_1 \\ \vdots \\ Y_n \end{pmatrix}$$

to be the vector of n response observations. Let

$$\boldsymbol{X} = \begin{pmatrix} 1 & X_{11} & X_{21} & \cdots & X_{1p} \\ 1 & X_{21} & X_{22} & \cdots & X_{2p} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & X_{n1} & X_{n2} & \cdots & X_{np} \end{pmatrix}$$

[4] https://www.thebalance.com/what-is-gdp-definition-of-gross-domestic-product-3306038

to be the vector of n ones plus n observations of p predictors. Then let

$$\boldsymbol{\beta} = \begin{pmatrix} \beta_1 \\ \vdots \\ Y\beta_n \end{pmatrix}$$

to be the vector of n coefficients. Finally, let

$$\boldsymbol{\epsilon} = \begin{pmatrix} \epsilon_1 \\ \vdots \\ \epsilon_n \end{pmatrix}$$

to be the vector of n independent white noises.

3.2.2 Formula Derivation

Consider this Least Squares problem (Weixi, Yijiang, and Professor Blais, unpublished):

$$\boldsymbol{\epsilon} = \boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta},$$

$$\begin{aligned} \min_{\boldsymbol{\beta}\in\mathbb{R}^{p+1}} ||\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}|| &= (\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta})^T(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}) \\ &= \boldsymbol{Y}^T\boldsymbol{Y} - 2(\boldsymbol{X}\boldsymbol{\beta})^T\boldsymbol{Y} + (\boldsymbol{X}\boldsymbol{\beta})^T\boldsymbol{X}\boldsymbol{\beta} \\ &= \boldsymbol{Y}^T\boldsymbol{Y} - 2\boldsymbol{\beta}^T\boldsymbol{X}^T\boldsymbol{Y} + \boldsymbol{\beta}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\beta} \end{aligned}$$

Set the 1-st order condition as 0

$$\bigtriangledown\boldsymbol{\beta} = 0$$

$\Leftrightarrow$

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{Y}.$$

Hence the predicted values are

$$\hat{Y} = \boldsymbol{X}\hat{\boldsymbol{\beta}} = X(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{Y}$$

Note: Here hat matrix (Ruppert, 2004, p. 195) $\hat{H}$ is defined as

$$\hat{H} = X(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T,$$

the leverage value H_{ii} is large when the predictors for observation i are atypical. General, if $H_{ii} > \frac{2(p+1)}{n}$, then it's a cause fro concern.

3.3 Analysis of Variance

The Analysis of Variance (ANOVA) can provide information about levels of variability within our regression model and form a basis for tests of significance (Weixi, Yijiang, and Professor Blais, unpublished).

We use the total sum of squared to describe the total variation in Y, the definition is shown as below.

Def 1. *The **total sum of squaresd** (SS) is defined as*

$$Totall\ SS = \sum_{i=1}^{n}(Y_i - \bar{Y})^2,$$

where Y_i is the i-th obervation of response Y, $\bar{Y}$ is the mean of all Y observations (Weixi, Yijiang, and Professor Blais, unpublished).

It contains two parts, a part which can be predicted by X and one cannot. The part of variance which can be predicted is called regression sum of squares.
Def 2. *The **regression sum of squares** is defined as*

$$Regression\ SS = \sum_{i=1}^{n}(\hat{Y}_i - \bar{Y})^2,$$

where $\hat{Y}_i$ is i-th predicted value (Weixi, Yijiang, and Professor Blais, unpublished).

The part which cannot be predicted is the residual error sum of squares.
Def 3. *The **residual error sum of squares** (SSE) is defined as*

$$SSE = \sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2,$$

(Weixi, Yijiang, and Professor Blais, unpublished).

Hence there is another way to represent total sum of squares is

$$\text{Total } SS = \text{Regression } SS + \text{Residual Rrror } SS.$$

Then we want to know how much of the total variance can be explained by our model, so we introduced R square.
Def 4. *The $\boldsymbol{R^2}$ is defined as*

$$R^2 = \frac{Regression\ SS}{Total\ SS} = 1 - \frac{SSE}{Total\ SS},$$

(Weixi, Yijiang, and Professor Blais, unpublished).

After we consider the Degrees of Freedom (Ruppert, 2004, p. 178), we introduced the Mean sum of squares which defined as its sum of squares divided by its degree of freedom.
Def 5. *The **mean sum of squares** (MS) is defined as*

$$MS = \frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{n - p - 1},$$

where n is the total number of observations, p is the number of predictors (Weixi, Yijiang, and Professor Blais, unpublished).

Since when add more predictors in building model, the R^2 must increse, it won't helpful for selecting model. Hence we introduce adjusted R^2 which can avoid this problem.

Def 6. *The **adjusted** R^2 is defined as*

$$\textit{Adjusted } R^2 = 1 - \frac{MS}{\frac{1}{n}\textit{Total SS}},$$

(Weixi, Yijiang, and Professor Blais, unpublished).

3.4 Model selection

We use several tools to help us to select good model.

3.4.1 Variance inflation factor

Variance inflation factor (VIF) (Weixi, Yijiang, and Professor Blais, unpublished) measures how much the squared standard error of that variable is raised by having the other predictor variables in the model. First, we regress predictor X_j on the other p-1 predictors, let R_j^2 be the R^2 of this regression, then the VIF_j is calculated as below:

$$VIF_j = \frac{1}{1 - R_j^2}.$$

The VIFs of the linear regression indicate the degree that the variances in the regression estimates are increased due to multicollinearity. The general rule of thumb is that VIFs exceeding 4 warrant further investigation, while VIFs exceeding 10 are signs of serious multicollinearity requiring correction[5].

3.4.2 Mallows' C_p

The Mallows' Cp can estimate how well a model predicts new responses (Weixi, Yijiang, and Professor Blais, unpublished).

Suppose we have M predictor variables, $\hat{\sigma}^2_{\epsilon M}$ estimate of σ^2_ϵ using all M predictors, where the estimate of σ is SSE divided by $(n - p - 1)$. Define $SSE(p)$ be sum of squared for residual error using $p \leq M$ predictors. Then

$$C_p = \frac{SSE(p)}{\hat{\sigma}^2_{\epsilon M}} - n + 2(p + 1)$$

3.4.3 Residuals

The **Raw residual** is defined as (Weixi, Yijiang, and Professor Blais, unpublished)

$$\hat{\epsilon}_i = Y_i - \hat{Y}_i.$$

The **Studentized residual** is defined as (Weixi, Yijiang, and Professor Blais, unpublished)

$$\frac{\hat{\epsilon}_i}{s\sqrt{1 - H_{ii}}},$$

[5] https://onlinecourses.science.psu.edu/stat501/node/347

where s is the estimate of σ_ϵ.

The **externally studentized residual** is defined as (Weixi, Yijiang, and Professor Blais, unpublished)

$$\frac{\hat{\epsilon}_i}{s(-i)\sqrt{1-H_{ii}}},$$

where $s(-i)$ is the estimate of σ_ϵ by fitting the model with the i-th observation removed.

Note: A rule of thumb is that a standardized or studentized residual is outlying if its absolute value exceeds 2 and extremely outlying if it exceeds 3 (Ruppert, 2004, p. 196).

The **Cook's Distance** is defined as (Weixi, Yijiang, and Professor Blais, unpublished)

$$\frac{\sum_{j=1}^{n}[\hat{Y}_j - \hat{Y}_j(-i)]^2}{(p+1)s^2},$$

where $\hat{Y}_j(-i)$ is the j-th fitted value using estimate of $\boldsymbol{\beta}$ obtained with the i-th observation removed.

Note: Some texts tell that points for which Cook's distance is higher than 1 are to be considered as influential. Other texts give a threshold of $\frac{4}{N}$ or $\frac{4}{N-p-1}$, where N is the number of observations and P is the number of predictors[6]. In the report, we will use $\frac{4}{N-p-1}$ as a threshold.

[6] https://en.wikipedia.org/wiki/Cook%27s_distance

Chapter 4

Implementation

4.1 Build model with all predictors

At the begining, we use all predictors to build model, we call it Model_all. By running our own function code as shown in Appendix A.1, we get coefficients as below:

```
beta_0: 6.96873188427
beta_1: -0.00889537806 1
beta_2: -0.0127132494073
beta_3: 1.73680748112
beta_4: -0.153909017825
beta_5: -0.178097147643
```

FIGURE 4.1: Coefficients of Model_all.

4.2 ANOVA Table

Then we run an analysis of variance (ANOVA) on Model_all, by using inner function code as shown in Appendix A.2, we get ANOVA table as below:

	df	sum_sq	mean_sq	F	PR(>F)
CPI	1.0	46.788630	46.788630	9.010923	0.003316
PPI	1.0	4.805026	4.805026	0.925390	0.338155
us_consumer	1.0	61.616827	61.616827	11.866654	0.000807
interest_rate	1.0	0.750872	0.750872	0.144609	0.704468
unemployment	1.0	2.781473	2.781473	0.535678	0.465772
Residual	111.0	576.360249	5.192435	NaN	NaN

FIGURE 4.2: ANOVA.

We can see from figure 4.2 that the P value of two predictors, which are CPI and us_consumer, less than 0.05, which means Model_all predicts the response variable better than the mean of the response. And by using our own function code as shown in Appendix A.3, we get adjusted R^2 of Model_all with 0.1309771. That means about 13.1% of the variability in the response is explained by all predictors. Although the regression is usefull since the model predicts the response variable better than the mean of the response, but the adjusted R^2 is small, so we are going to use other tools to pick a better model.

4.3 VIF

Now, we compute the VIF for each predictor, by using our own function code as shown in Appendix A.4, we get result as below:

```
CPI: 58.0119891999
PPI: 46.4407486374
us_consume: 1.13047097678
interest_rate: 6.29195439364
unemployment: 2.97197043456
```

FIGURE 4.3: VIF.

From this result, we decide to remove predictor CPI and predictor PPI, since those VIF values all greater than 10. Although the VIF of predictor interest_rate is 6.29, we can keep it at this time, after using other tools to decide whether we need remove this predictor.

4.4 Mallows' C_p and adjusted R^2

Then we compute C_p values of models that built by all possible combinations of remaining predictors. By using our own function code as shown in Appendix A.5, we get result as below:

Value	us_consume	unemployment	interest_rate	1*2	1*3	2*3
C_p	1.96102195504	14.8112762274	12.3291294201	2.71920333545	2.26372641143	13.7909896817
adj_r^2	0.121014738155	0.0227624442532	0.0417407938209	0.122882502139	0.126395600791	0.0374856891819

FIGURE 4.4: C_p and R^2, 1*2 means the model built by us_consumer and unemployment, 1*3 means the model built by us_consume and interest_rate and 2*3 means the model built by unemployment and interest_rate.

First we look at C_p value, we can conclude:

- the model contains only the predictor us_consumer and its C_p value is 1.96. The C_p is less than p + 1 = 2, it suggests the model is unbiased.
- the model contains the predictors us_consumer and unemployment and its C_p value is 2.7. The C_p is less than p + 1 =3, it suggests the model is unbiased.
- the model contains the predictors us_consumer and interest_rate and its C_p value is 2.26. The C_p is less than p + 1 =3, it suggests the model is unbiased.

Then we look at adjusted R^2, we can see the model with predictors us_consume and interest_rate is the best since its adjusted R^2 is largest among three.

4.5 Leverage

Now we look at leverage values. By using our own function code as shown in Appendix A.6, we get results as below:

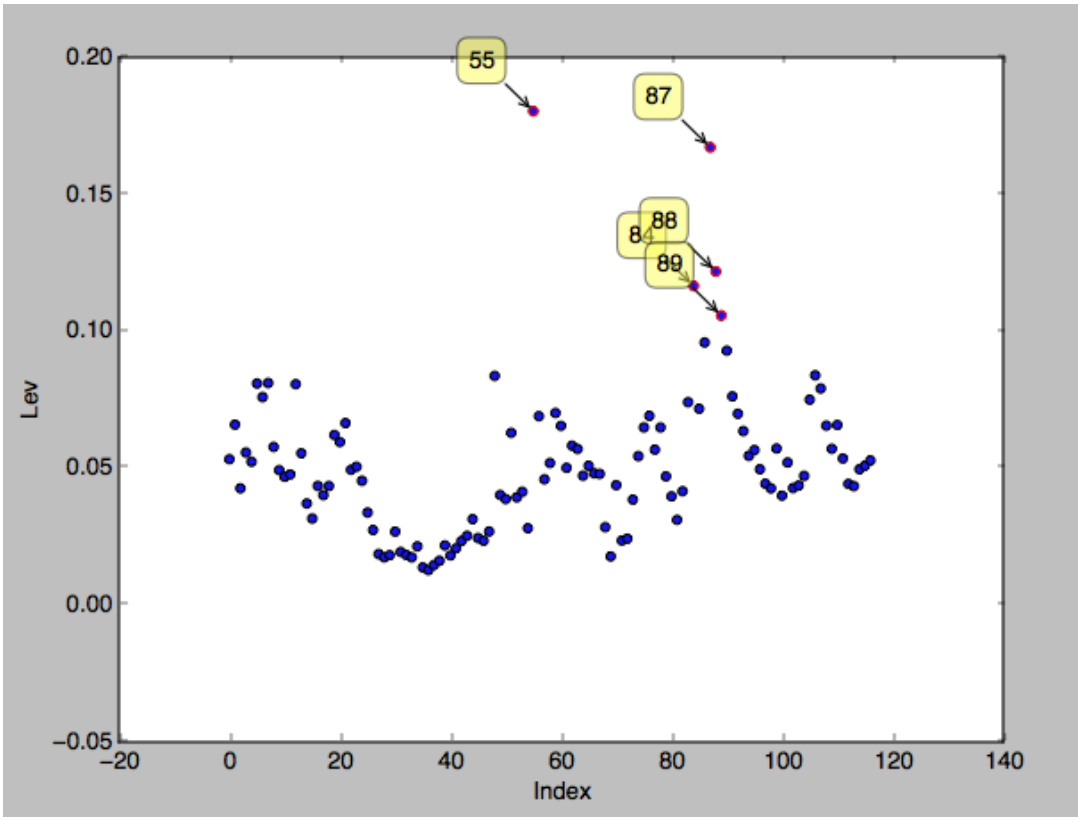


FIGURE 4.5: Scatter plot of leverage

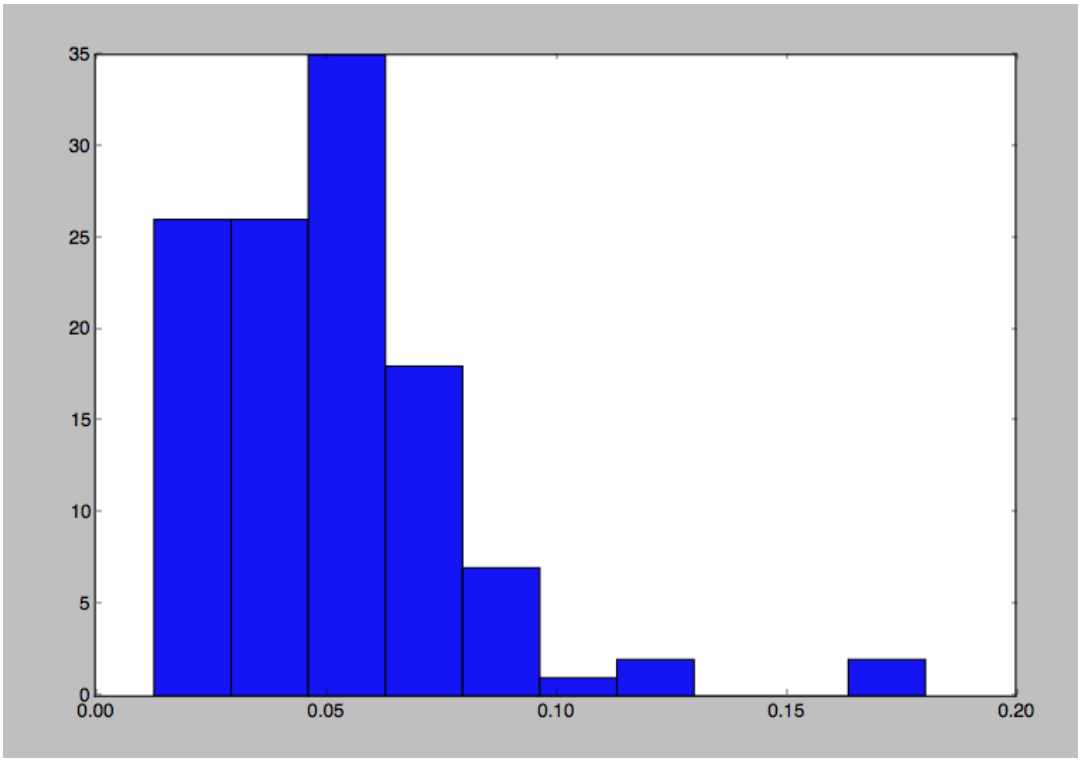


FIGURE 4.6: Hist of leverage

As shown in figure 4.5, there are leverage values greater than $\frac{2(p+1)}{n}$, which means these points have concerning leverage values.

4.6 Residual

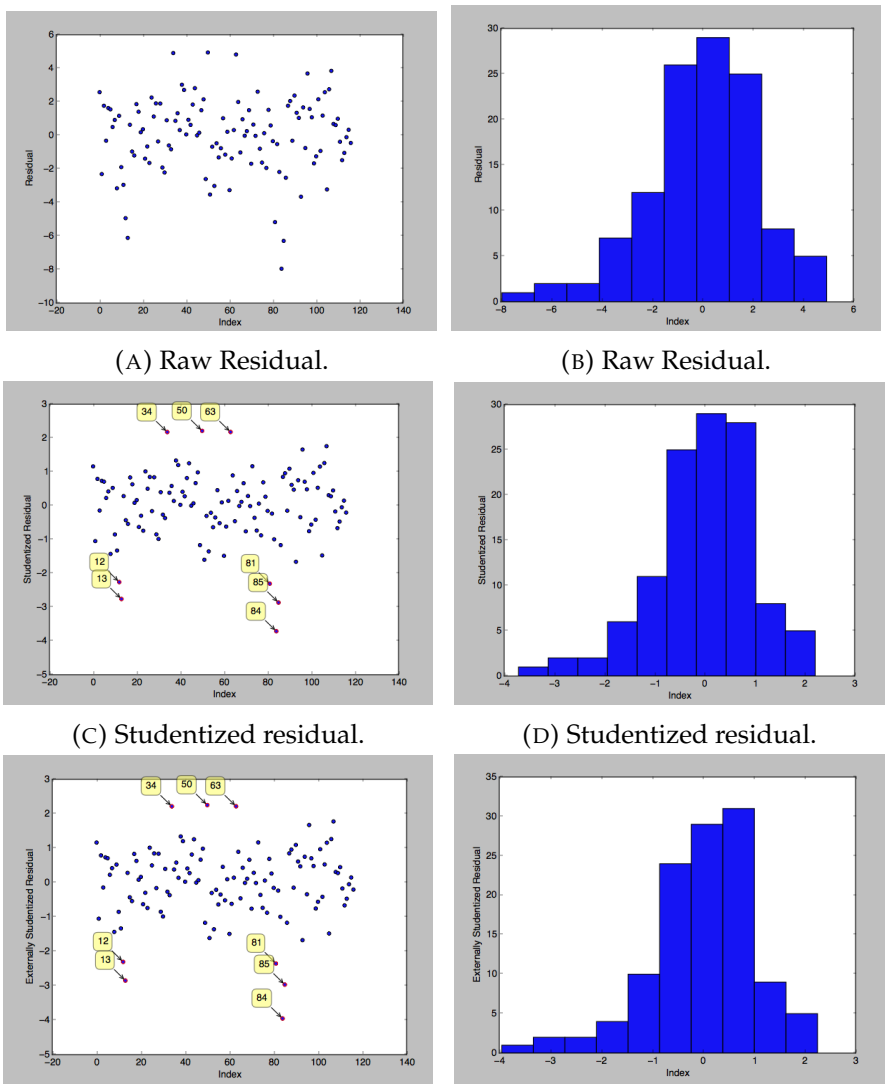
We will present 3 types of residual. By using our own function code as shown in Appendix A.7, we get results as shown in figure 4.7. We can see from Studentized residual or Externallu studentized residual, they all show about 8 points may be outliers, and one of them is textremely outlier, which is the 84-th observation.

4.7 Cook's Distance

Now we use Cook's Distance to check whether there are exist points that may be influential. By using our own function code as shown in Appendix A.8, we get results as shown in figure 4.8. We can conclude there is at least one case, which is 84-th observation, would be a cause for concern.

4.8 Choose Model

From the above analysis, firstly, we decide to eliminate outliers, we tried to remove some combinations of points that may be influential with our model, those points



(A) Raw Residual. (B) Raw Residual.

(C) Studentized residual. (D) Studentized residual.

(E) Externally Studentized residual. (F) Externally Studentized residual.

FIGURE 4.7: 3 types of residual.

come from our Cook's Distance. We found when we removed the 12 point from our dataset, we got a higher adjusted R^2. The result is as shown in figure 4.9.

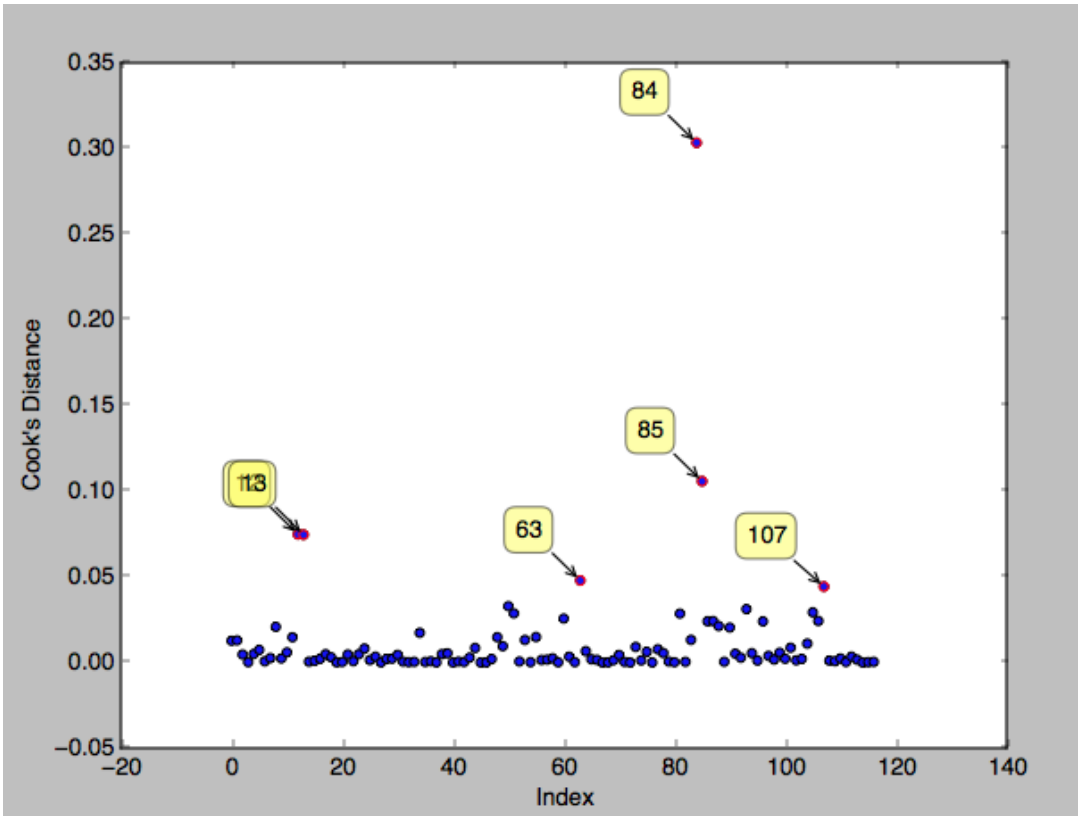


FIGURE 4.8: Cook's Distance

```
beta_0: 6.78103428731
beta_1: -0.0128086620564
beta_2: -0.00722429121492
beta_3: 1.7305238011
beta_4: -0.133133692629
beta_5: -0.172833677063
0.136956650005
                    sum_sq     df          F    PR(>F)
CPI               0.462544    1.0   0.089702  0.765121
PPI               0.107845    1.0   0.020915  0.885276
us_consumer      58.316157    1.0  11.309373  0.001062
interest_rate     2.377562    1.0   0.461086  0.498543
unemployment      2.618801    1.0   0.507869  0.477572
Residual        567.208911  110.0        NaN       NaN
CPI: 57.3959861434
PPI: 46.3137785024
us_consume: 1.12871794452
interest_rate: 6.17699634095
unemployment: 2.9717002396
+-----------+---------------+-----------------+-----------------+-----------------+-----------------+-----------------+
| Value     |   us_consume  |   unemployment  |  interest_rate  |       1*2       |       1*3       |       2*3       |
+-----------+---------------+-----------------+-----------------+-----------------+-----------------+-----------------+
| C_p       | 2.45310606191 |  15.7245620352  |  12.2194546988  | 3.17232163466   | 2.15892123044   |  13.8393835818  |
| adj_r^2   | 0.125063728962| 0.0236101415487 | 0.0504049193229 | 0.127198526665  | 0.13501401859   | 0.0449325869978 |
+-----------+---------------+-----------------+-----------------+-----------------+-----------------+-----------------+
```

FIGURE 4.9: Highest R^2.

From ANOVA, we know this model predicts the response variable better than the mean of the response, we choose the model with predictors us_consumer and interest_rate. And from VIF, we can remove predictors CPI and PPI, since their VIFs much greater than 10. Then, by choosing a appropriate C_P values, higher adjusted R^2 and lower MSE, we choose the model with predictors us_consumerHW and interest_rate (Thees procedures are very same as we presented before).

After choose our model, we are going to compute its coefficients. By using our own function code as shown in Appendix A.9, the result is:

```
beta_0: 1.39709378406
beta_1: 1.78554543949
beta_2: 0.126836658619
```

FIGURE 4.10: Coefficients of two predictors model.

Now, we are comparing this model to Model_all, recall that the adjusted R^2 of Model_all is 0.1309771 while our new model's adjusted R^2 is 0.135014, which is greater than old model. That means new model's predictors can explain more percentage of the variability in the response than ole one.

4.9 Predict 2017 GDP growth rates

By using our own function as shown in Appendix A.10, our result is:

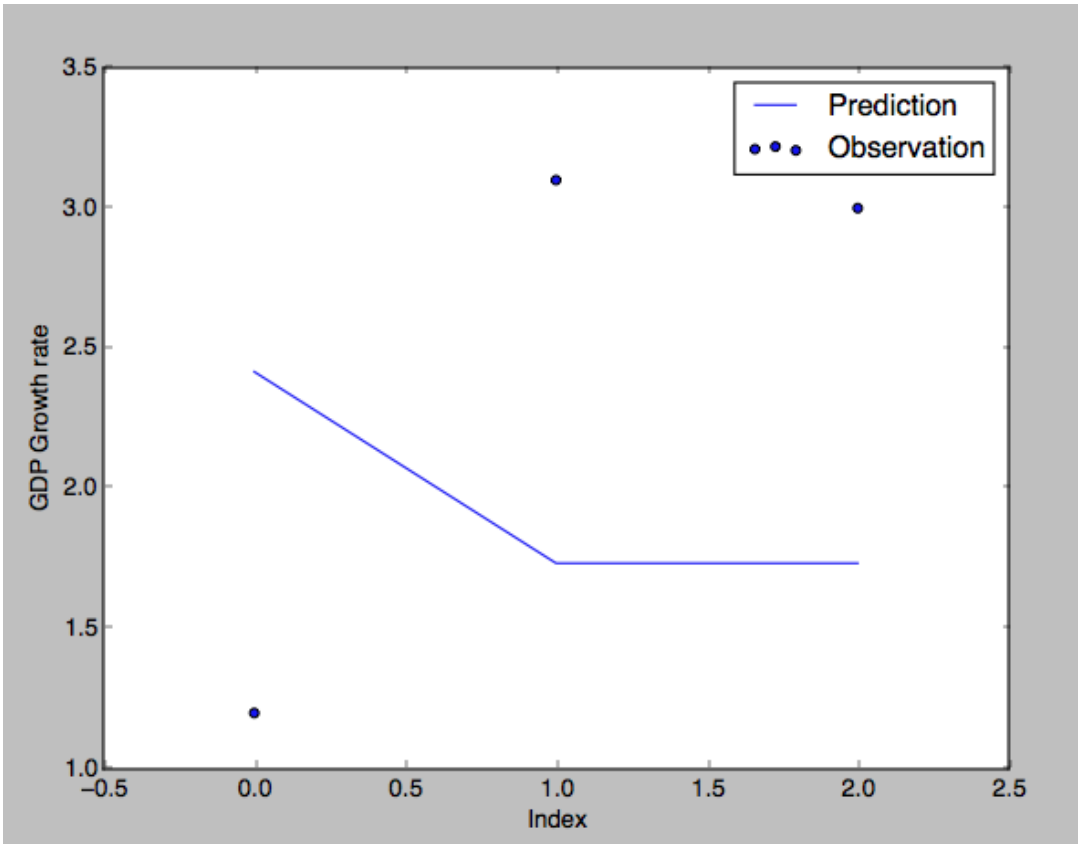


FIGURE 4.11: Predictions v.s. real values.

4.10 Conclusion

- Our GDP growth rate is real GDP, which exclude the influence of inflation. Hence in VIF process, our two measuring inflation predictors CPI and PPI was been removed.
- Our predictors is very less as well as our observations. That leads to those models built by any combinations of all predictors can not explain the response very well, in other words, the adjusted R^2s are very low which close to 10%. On the other hand, since the predictors we collected are not very correlated with GDP but US Consumer Spending, it also leads the real GDP growth rate cannot be explained very well by our predictors.
- Also since the observations are not enough, we cannot train the model very well, and the correlation between predictors and response is actually not linear, so the predictions are not precise.
- And our tools can help us to find a good model with good MSE but they cannot ensure this model has a good test MSE.

Appendix A

Code

A.1 Calculate Model_all coefficients

```
    def calCoeff(self, response, predictors, *delete):
        if len(delete) == 0:
            X = []

            ## Construct X
            for i in range(len(predictors)):
                X.append(predictors[i])
            X = np.transpose(X)
            X = np.concatenate((np.ones([len(X), 1]), X), axis=1)

            ## Construct Y
            Y = np.transpose(response)
            self.Y = Y

            ## Calculate beta
            beta = np.dot(np.dot(np.linalg.inv(np.dot(np.transpose(
                X), X)), np.transpose(X)), Y)
            self.beta = beta

            ## Calculate Hat H
            self.H = np.dot(np.dot(X, np.linalg.inv(np.dot(np.
                transpose(X), X))), np.transpose(X))

            ## Calculate predict Y
            self.predict_Y = np.dot(X, beta)
        else:
            X = []

            ## Construct X
            predictors = np.delete(predictors, delete, axis = 1)
            for i in range(len(predictors)):
                X.append(predictors[i])
            X = np.transpose(X)
            X = np.concatenate((np.ones([len(X), 1]), X), axis=1)

            ## Construct Y
            response = np.delete(response, delete)
            Y = np.transpose(response)
            self.Y = Y

            ## Calculate beta
```

```
            beta = np.dot(np.dot(np.linalg.inv(np.dot(np.transpose(
                X), X)), np.transpose(X)), Y)
            self.beta = beta

            ## Calculate Hat H
            self.H = np.dot(np.dot(X, np.linalg.inv(np.dot(np.
                transpose(X), X))), np.transpose(X))

            ## Calculate predict Y
            self.predict_Y = np.dot(X, beta)

    def all_predictors_model(self):
        predictors = self.value[1:]
        response = self.value[0]
        self.calCoeff(response, predictors)
        print("beta_0: " + str(self.beta[0]))
        print("beta_1: " + str(self.beta[1]))
        print("beta_2: " + str(self.beta[2]))
        print("beta_3: " + str(self.beta[3]))
        print("beta_4: " + str(self.beta[4]))
        print("beta_5: " + str(self.beta[5]))
```

A.2 ANOVA

```
    def ANOVA_innerFc(self):
        df = pd.DataFrame({'GDP': self.value[0],
                           'CPI': self.value[1],
                           'PPI': self.value[2],
                           'us_consumer': self.value[3],
                           'interest_rate': self.value[4],
                           'unemployment': self.value[5]})
        model = ols('GDP ~ CPI+PPI+us_consumer+interest_rate+
            unemployment', data = df).fit()
        aov_table = sm.stats.anova_lm(model, typ = 2)
        print(aov_table)
```

A.3 Model_all adjusted R^2

```
    def ANOVA_mannual(self):
        predictors = self.value[1:]
        response = self.value[0]
        self.calCoeff(response, predictors)
        total_SS = sum((self.Y - np.mean(self.Y))**2)
        regression_SS = sum((self.predict_Y - np.mean(self.Y))**2)
        error_SS = sum((self.Y - self.predict_Y)**2)
        error_MS = error_SS / (len(self.Y) - len(self.beta) + 1 -
            1)
        total_MS = total_SS / (len(self.Y) - 1)
        r_squared = 1 - error_SS / total_SS
        adj_r_squared = 1 - error_MS / total_MS
        print(adj_r_squared)
```

A.4 VIF

```
    def VIF(self):
        VIF_all = []
        for i in range(len(self.value) - 1):
            predictors = self.value[1:]
            response = predictors[i]
            predictors = np.delete(predictors, i, axis = 0)
            self.calCoeff(response, predictors)
            total_SS = sum((self.Y - np.mean(self.Y)) ** 2)
            error_SS = sum((self.Y - self.predict_Y) ** 2)
            r_squared = 1 - error_SS / total_SS
            VIF_all.append(1/(1 - r_squared))
        print("CPI: " + str(VIF_all[0]))
        print("PPI: " + str(VIF_all[1]))
        print("us_consume: " + str(VIF_all[2]))
        print("interest_rate: " + str(VIF_all[3]))
        print("unemployment: " + str(VIF_all[4]))
```

A.5 C_p

```
    def Cp(self):
        adj_r_squared = ["adj_r^2"]
        C_p_all = ["C_p"]
        ## Using only us_consumer predictor
        predictors = self.value[1:]
        predictors = np.delete(predictors, [0, 1], axis=0)
        response = self.value[0]
        self.calCoeff(response, predictors)
        residual = self.Y - self.predict_Y
        variance_epsilon = sum((residual - np.mean(residual)) ** 2)
            / (len(self.Y) - len(predictors) - 1)
        predictors = np.asarray([self.value[3]])
        response = self.value[0]
        self.calCoeff(response, predictors)
        error_SS = sum((self.Y - self.predict_Y) ** 2)
        C_p = error_SS/variance_epsilon - len(self.Y) + 2*(len(
            predictors) + 1)
        C_p_all.append(C_p)
        # R^2
        error_MS = error_SS / (len(self.Y) - len(self.beta) + 1 -
            1)
        total_SS = sum((self.Y - np.mean(self.Y)) ** 2)
        total_MS = total_SS / (len(self.Y) - 1)
        adj_r_squared.append(1 - error_MS / total_MS)

        ## Using only unemployment rate predictor
        predictors = self.value[1:]
        predictors = np.delete(predictors, [0, 1], axis=0)
        response = self.value[0]
        self.calCoeff(response, predictors)
        residual = self.Y - self.predict_Y
        variance_epsilon = sum((residual - np.mean(residual)) ** 2)
            / (len(self.Y) - len(predictors) - 1)
```

```
30          predictors = np.asarray([self.value[5]])
31          response = self.value[0]
32          self.calCoeff(response, predictors)
33          error_SS = sum((self.Y - self.predict_Y) ** 2)
34          C_p = error_SS / variance_epsilon - len(self.Y) + 2 * (len(
                predictors) + 1)
35          C_p_all.append(C_p)
36          # R^2
37          error_MS = error_SS / (len(self.Y) - len(self.beta) + 1 -
                1)
38          total_SS = sum((self.Y - np.mean(self.Y)) ** 2)
39          total_MS = total_SS / (len(self.Y) - 1)
40          adj_r_squared.append(1 - error_MS / total_MS)
41
42          ## Using only interest rate predictor
43          predictors = self.value[1:]
44          predictors = np.delete(predictors, [0, 1], axis=0)
45          response = self.value[0]
46          self.calCoeff(response, predictors)
47          residual = self.Y - self.predict_Y
48          variance_epsilon = sum((residual - np.mean(residual)) ** 2)
                / (len(self.Y) - len(predictors) - 1)
49          predictors = np.asarray([self.value[4]])
50          response = self.value[0]
51          self.calCoeff(response, predictors)
52          error_SS = sum((self.Y - self.predict_Y) ** 2)
53          C_p = error_SS / variance_epsilon - len(self.Y) + 2 * (len(
                predictors) + 1)
54          C_p_all.append(C_p)
55          # R^2
56          error_MS = error_SS / (len(self.Y) - len(self.beta) + 1 -
                1)
57          total_SS = sum((self.Y - np.mean(self.Y)) ** 2)
58          total_MS = total_SS / (len(self.Y) - 1)
59          adj_r_squared.append(1 - error_MS / total_MS)
60
61          ## Using us_consume and umemployment
62          predictors = self.value[1:]
63          predictors = np.delete(predictors, [0, 1], axis = 0)
64          response = self.value[0]
65          self.calCoeff(response, predictors)
66          residual = self.Y - self.predict_Y
67          variance_epsilon = sum((residual - np.mean(residual)) ** 2)
                / (len(self.Y) - len(predictors) - 1)
68          predictors = self.value[1:]
69          predictors = np.delete(predictors, [0, 1, 3], axis=0)
70          response = self.value[0]
71          self.calCoeff(response, predictors)
72          error_SS = sum((self.Y - self.predict_Y) ** 2)
73          C_p = error_SS / variance_epsilon - len(self.Y) + 2 * (len(
                predictors) + 1)
74          C_p_all.append(C_p)
75          # R^2
76          error_MS = error_SS / (len(self.Y) - len(self.beta) + 1 -
                1)
77          total_SS = sum((self.Y - np.mean(self.Y)) ** 2)
78          total_MS = total_SS / (len(self.Y) - 1)
```

```
79          adj_r_squared.append(1 - error_MS / total_MS)
80
81          ## Using us_consume and interest_rate
82          predictors = self.value[1:]
83          predictors = np.delete(predictors, [0, 1], axis=0)
84          response = self.value[0]
85          self.calCoeff(response, predictors)
86          residual = self.Y - self.predict_Y
87          variance_epsilon = sum((residual - np.mean(residual)) ** 2)
                / (len(self.Y) - len(predictors) - 1)
88          predictors = self.value[1:]
89          predictors = np.delete(predictors, [0, 1, 4], axis=0)
90          response = self.value[0]
91          self.calCoeff(response, predictors)
92          error_SS = sum((self.Y - self.predict_Y) ** 2)
93          C_p = error_SS / variance_epsilon - len(self.Y) + 2 * (len(
               predictors) + 1)
94          C_p_all.append(C_p)
95          # R^2
96          error_MS = error_SS / (len(self.Y) - len(self.beta) + 1 -
               1)
97          total_SS = sum((self.Y - np.mean(self.Y)) ** 2)
98          total_MS = total_SS / (len(self.Y) - 1)
99          adj_r_squared.append(1 - error_MS / total_MS)
100
101         ## Using unemployment and interest_rate
102         predictors = self.value[1:]
103         predictors = np.delete(predictors, [0, 1], axis=0)
104         response = self.value[0]
105         self.calCoeff(response, predictors)
106         residual = self.Y - self.predict_Y
107         variance_epsilon = sum((residual - np.mean(residual)) ** 2)
                / (len(self.Y) - len(predictors) - 1)
108         predictors = self.value[1:]
109         predictors = np.delete(predictors, [0, 1, 2], axis=0)
110         response = self.value[0]
111         self.calCoeff(response, predictors)
112         error_SS = sum((self.Y - self.predict_Y) ** 2)
113         C_p = error_SS / variance_epsilon - len(self.Y) + 2 * (len(
               predictors) + 1)
114         C_p_all.append(C_p)
115         # R^2
116         error_MS = error_SS / (len(self.Y) - len(self.beta) + 1 -
               1)
117         total_SS = sum((self.Y - np.mean(self.Y)) ** 2)
118         total_MS = total_SS / (len(self.Y) - 1)
119         adj_r_squared.append(1 - error_MS / total_MS)
120
121         x = PrettyTable(
122             ["Value", "us_consume", "unemployment", "interest_rate"
                   , "1*2", "1*3", "2*3"])
123         x.align["Value"] = "l"
124         x.padding_width = 1
125         x.add_row(C_p_all)
126         x.add_row(adj_r_squared)
127         print(x)
```

A.6 Leverage

```
    def leverage(self):
        predictors = self.value[1:]
        response = self.value[0]
        self.calCoeff(response, predictors)
        H_ii = [self.H[i][i] for i in range(len(self.Y))]
        leverage_X = [j for j in range(len(H_ii)) if H_ii[j] > 2*(
            len(predictors) + 1) / len(self.Y)]
        leverage_Y = [H_ii[j] for j in range(len(H_ii)) if H_ii[j]
             > 2*(len(predictors) + 1) / len(self.Y)]
        Xaxis = np.arange(len(H_ii))
        plt.scatter(Xaxis, H_ii)
        labels = ['{0}'.format(i) for i in leverage_X]
        plt.scatter(leverage_X, leverage_Y, edgecolors="red")
        for label, x, y in zip(labels, leverage_X, leverage_Y):
            plt.annotate(
                label,
                xy=(x, y), xytext=(-20, 20),
                textcoords='offset points', ha='right', va='bottom
                    ',
                bbox=dict(boxstyle='round,pad=0.5', fc='yellow',
                    alpha=0.5),
                arrowprops=dict(arrowstyle='->', connectionstyle='
                    arc3,rad=0'))

        plt.xlabel("Index")
        plt.ylabel("Lev")
        #plt.hist(H_ii)
        plt.show()
```

A.7 3 types of residual

```
    def rawResidual(self):
        predictors = self.value[1:]
        response = self.value[0]
        self.calCoeff(response, predictors)
        residual = self.Y - self.predict_Y
        #plt.scatter(np.arange(len(self.Y)), residual)
        plt.hist(residual)
        plt.xlabel("Index")
        plt.ylabel("Residual")
        plt.show()

    def studentizedResidual(self):
        predictors = self.value[1:]
        response = self.value[0]
        self.calCoeff(response, predictors)
        residual = self.Y-self.predict_Y
        variance_epsilon = sum((residual - np.mean(residual)) ** 2)
            / (len(self.Y) - len(predictors) - 1)
        H_ii = [self.H[i][i] for i in range(len(self.Y))]
        s_residual = residual / (np.sqrt(variance_epsilon * (1 - np
            .asarray(H_ii))))
```

```
20          plt.scatter(np.arange(len(s_residual)), s_residual)
21          #plt.hist(s_residual)
22          plt.xlabel("Index")
23          plt.ylabel("Studentized Residual")
24          plt.show()
25
26      def exterStudentizedResidual(self):
27          predictors = self.value[1:]
28          response = self.value[0]
29          self.calCoeff(response, predictors)
30          residual = self.Y - self.predict_Y
31          H_ii = [self.H[i][i] for i in range(len(self.Y))]
32
33          variance_epsilon = []
34          ## remove i_th observation
35          for i in range(len(self.Y)):
36              response = self.value[0]
37              self.calCoeff(response, predictors, i)
38              #predictors = self.value[1:]
39              # X = []
40              # ## Construct X
41              # for i in range(len(predictors)):
42              #     X.append(predictors[i])
43              # X = np.transpose(X)
44              # X = np.concatenate((np.ones([len(X), 1]), X), axis=1)
45              # self.predict_Y = np.dot(X, self.beta)
46              new_residual = self.Y - self.predict_Y
47              variance_epsilon.append(sum((new_residual - np.mean(
                    new_residual)) ** 2) / (len(self.Y) - len(predictors
                    ) - 1))
48          e_s_residual = residual / (np.sqrt(variance_epsilon * (1 -
                np.asarray(H_ii))))
49          #plt.hist(e_s_residual)
50          plt.scatter(np.arange(len(e_s_residual)), e_s_residual)
51          plt.xlabel("Index")
52          plt.ylabel("Externally Studentized Residual")
53          plt.show()
```

A.8 Cook's Distance

```
1       def cookD(self):
2           predictors = self.value[1:]
3           response = self.value[0]
4           self.calCoeff(response, predictors)
5           residual = self.Y - self.predict_Y
6           variance_epsilon = sum((residual - np.mean(residual)) **
                2) / (len(self.Y) - len(predictors) - 1)
7           self.predict_Y_1 = self.predict_Y
8           cook_d = []
9           ## Remove i_th observation
10          for i in range(len(self.Y)):
11              self.calCoeff(response, predictors, i)
12              beta = self.beta
13              X = []
14              ## Construct X
```

```
            for i in range(len(predictors)):
                X.append(predictors[i])
            X = np.transpose(X)
            X = np.concatenate((np.ones([len(X), 1]), X), axis=1)
            self.predict_Y_2 = np.dot(X, beta)
            cook_d.append(sum((self.predict_Y_1 - self.predict_Y_2
                )**2) / ((len(predictors) + 1)*variance_epsilon))
        plt.scatter(np.arange(len(cook_d)), cook_d)
        plt.show()
```

A.9 pickedModel

```

    def pickedModel(self):
        predictors = self.value[1:]
        predictors = np.delete(predictors, [0, 1, 4], axis=0)
        response = self.value[0]
        self.calCoeff(response, predictors)
        print("beta_0: " + str(self.beta[0]))
        print("beta_1: " + str(self.beta[1]))
        print("beta_2: " + str(self.beta[2]))
```

A.10 Prediction

```
    def predict_(self):
        predictors = self.value[1:]
        predictors = np.delete(predictors, [0, 1, 4], axis=0)
        response = self.value[0]
        self.calCoeff(response, predictors)
        beta = self.beta
        X = []
        ## Construct X
        predictors = self.predict[1:]
        predictors = np.delete(predictors, [0, 1, 4], axis=0)
        for i in range(len(predictors)):
            X.append(predictors[i])
        X = np.transpose(X)
        X = np.concatenate((np.ones([len(X), 1]), X), axis=1)
        self.predict_Y_2 = np.dot(X, beta)
        Xaxis = np.arange(len(self.predict[0]))
        plt.scatter(Xaxis, self.predict[0], label = "Observation")
        plt.plot(Xaxis, self.predict_Y_2, label = "Prediction")
        plt.legend()
        plt.xlabel("Index")
        plt.ylabel("GDP Growth rate")
        plt.show()
```

Bibliography

Ruppert, D. (2004). *Statistics and Finance: An Introduction*. Springer Texts in Statistics. Springer. ISBN: 9780387202709.