# Bias – Variance Practice

**Optimization and Statistical Inference Lab**

김낙일 석사과정

# MobaXterm 접속

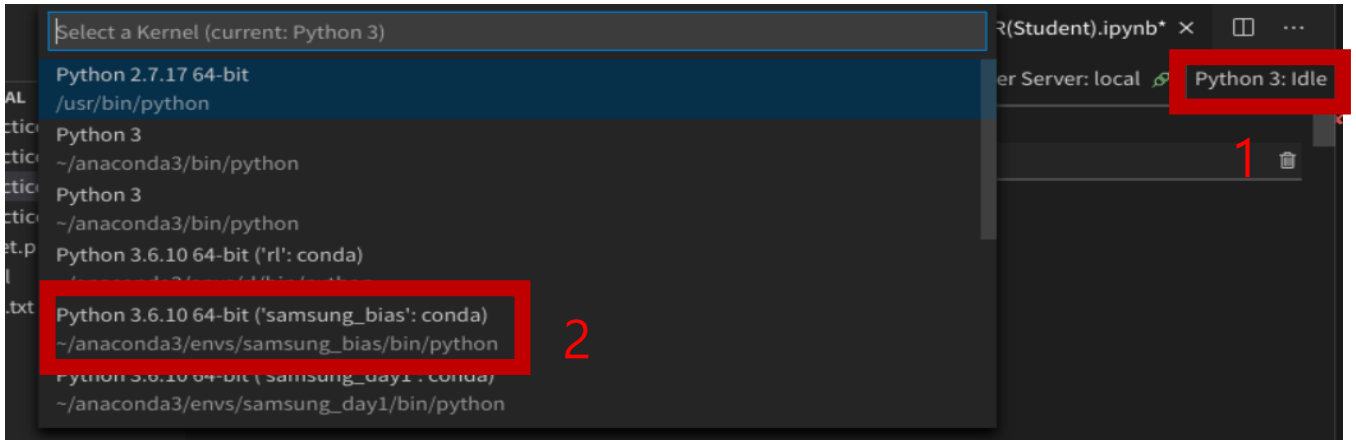
```shell
ssh -X -p 2222 com**02@143.248.159.**
source ~/.bashrc
```

Shell

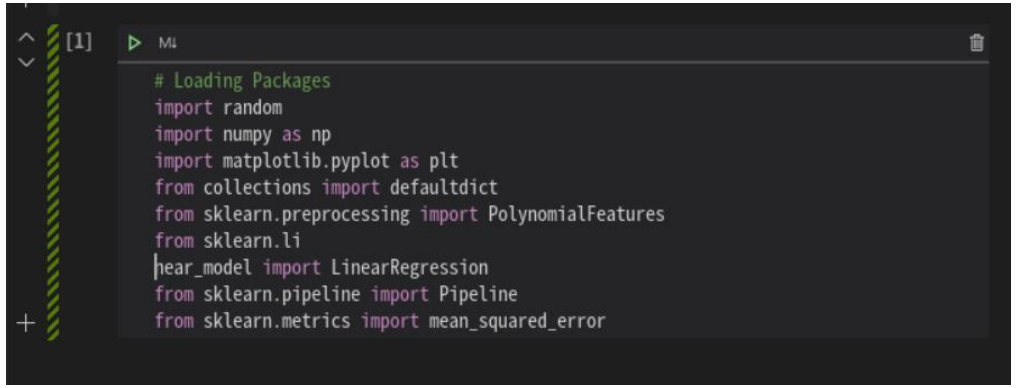
- PC 앞에 적혀진 IP 주소 접속
- 대문자 X 주의!

# 개발환경 설정

1. Terminal 창에 code : 입력! (영어 code 치고 뛰어쓰기 두번)
   * 실행 안될 시에 killall code 입력 후 실행
2. 실행된 visual studio code에서 왼쪽 위 File 탭 -> Open Folder -> Samsung_ds_tutorial 폴더로 이동 후 왼쪽 아래 OK 클릭
3. Bias-Variance-practice_KOR(Student).ipynb 실행 + 환경 바꿔주기 (그림 참고)
4. 패키지 import 되는 지 확인 (shift+enter)




가상환경 변경




패키지 import

# Git Clone, 가상환경 설치 (설치가 안되어 있을 경우)

```shell
git clone https://github.com/forestnoobie/samsung_ds_tutorial.git
cd samsung_ds_tutorial
conda env create --file samsung_bias.yml
conda activate samsung_bias
python -m ipykernel install --user --name=samsung_bias
jupyter notebook --port 9999
```

Shell

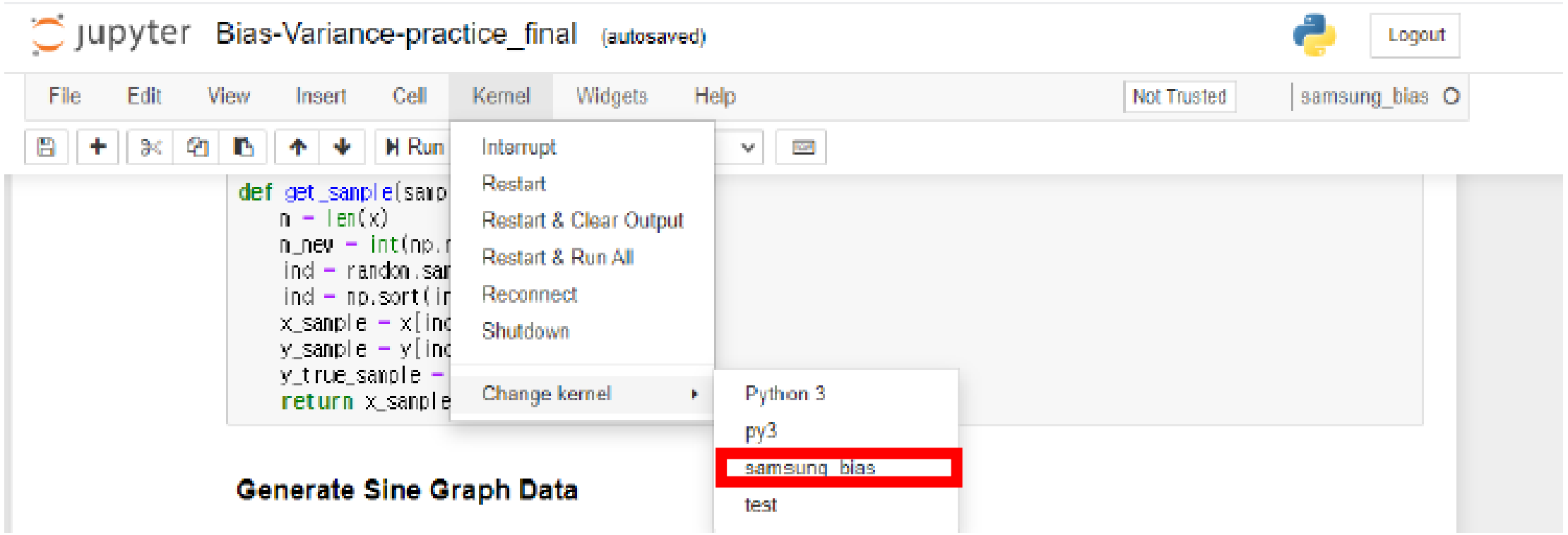
# Jupyter 및 Firefox 오류

Copy to clipboard

```shell
## Fire Fox 켜져 있을 경우
pidof firefox
# 켜져있으면 Process ID (PID)가 나옴
kill -9 [PID]

## jupyter server가 켜져있을 경우
jupyter notebook list
jupyter notebook stop [PORT]
# ex) 8888 사용 중이면 jupyter notebook stop 8888
# 위 명령어로 대처가 안 될 시
lsof -i tcp:[PORT]
kill -9 [PID]
```

Shell

# Jupyter notebook kernel 바꿔주기 (samsung_bias)

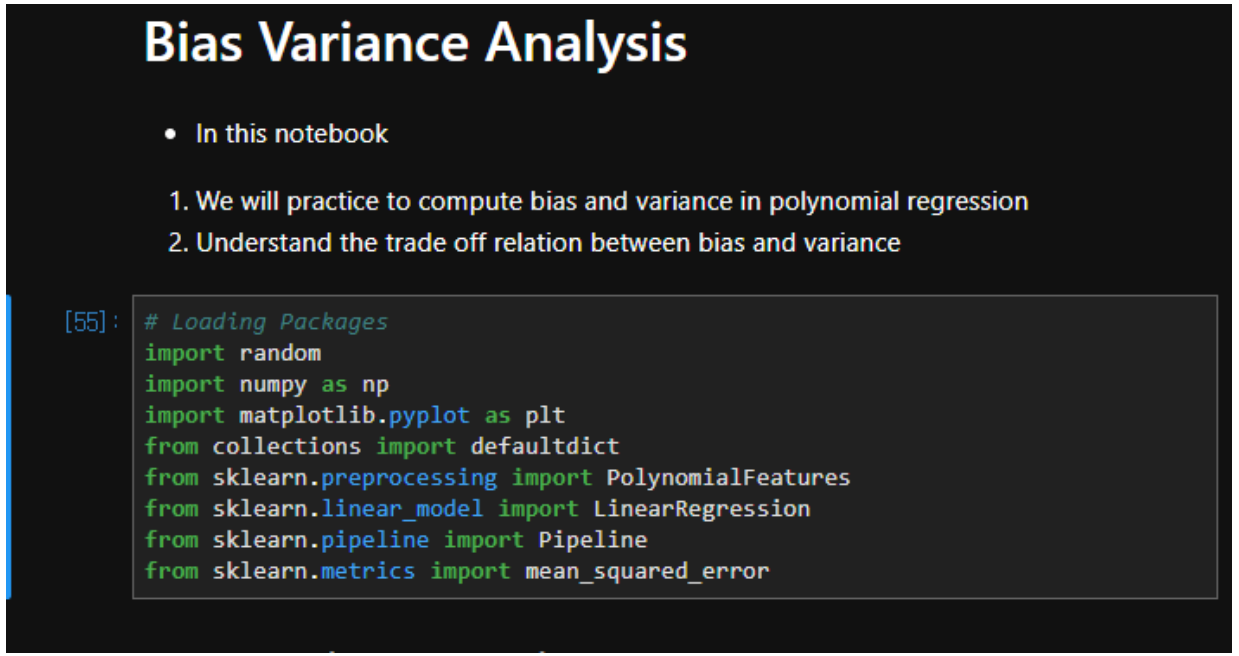


# Jupyter notebook Import

[1]:
```
# Loading Packages
import random
import numpy as np
import matplotlib.pyplot as plt
from collections import defaultdict
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.pipeline import Pipeline
from sklearn.metrics import mean_squared_error
```

# Jupyter Notebook 이란?

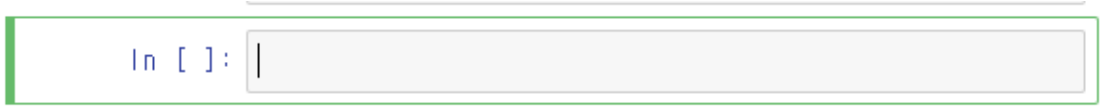
- 오픈소스 웹 어플리케이션
- 파이썬 코드 실행, 텍스트 작성 가능
- Cell 들로 구성



# Jupyter Notebook 이해

- 2가지 Mode 존재

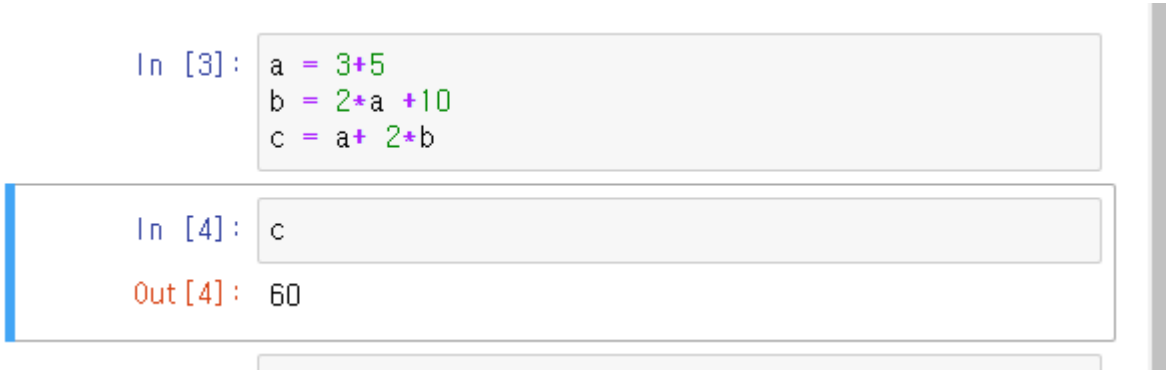

**Command Mode**
- 진입 방법 : [Esc]
- Cell 실행 : Ctrl + [Enter]
- Cell 합치기 : Shift + m
- Cell 추가하기 : b
- 변수 초기화 : 00




**Edit Mode**
- 진입 방법 : [Enter]
- 코드 작성

# Jupyter Notebook 이해

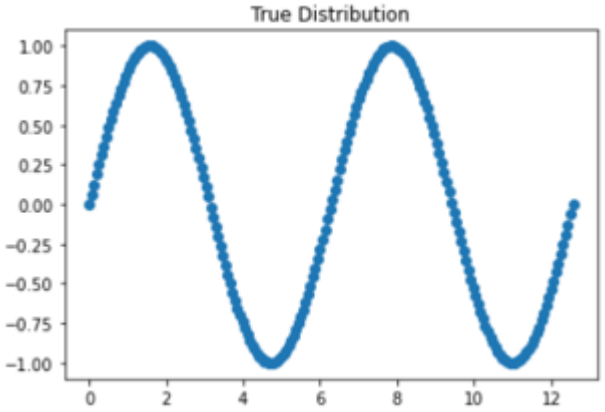
- 변수 안에 값을 바로 출력해볼 수 있다!
- 실습 중간에 **변수의 값을 확인**하고 싶으면 **아래와 같이 출력을 권장**

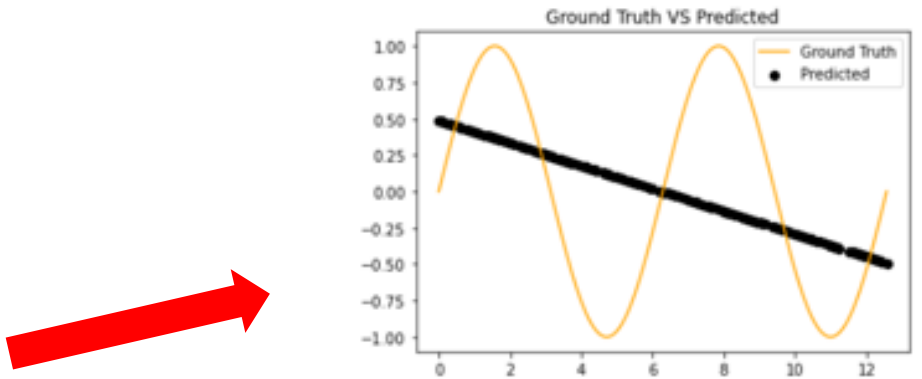


# Jupyter Notebook 정리

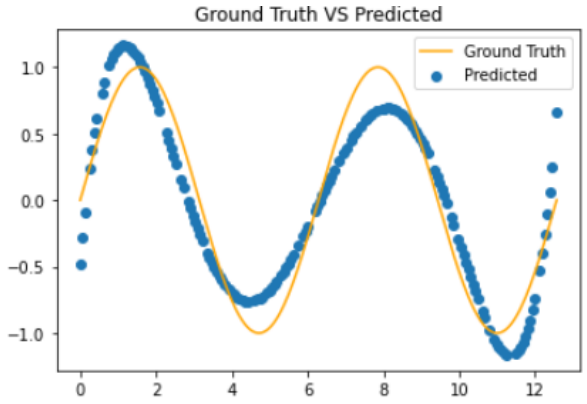
1. b : 셀 추가!
2. Ctrl + Enter : 실행!

# 실습 목표 : Polynomial Regression 을 Sin 함수 근사, Bias/Variance 비교




Bias? Variance?




Bias? Variance?

# 패키지 불러오기

[55]:
```
# Loading Packages
import random
import numpy as np
import matplotlib.pyplot as plt
from collections import defaultdict
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.pipeline import Pipeline
from sklearn.metrics import mean_squared_error
```

Random : 난수 생성에 관한 패키지

Numpy : 행렬을 다루기 위한 패키지

Matplotlib : 그래프 그리기 위한 시각화 패키지

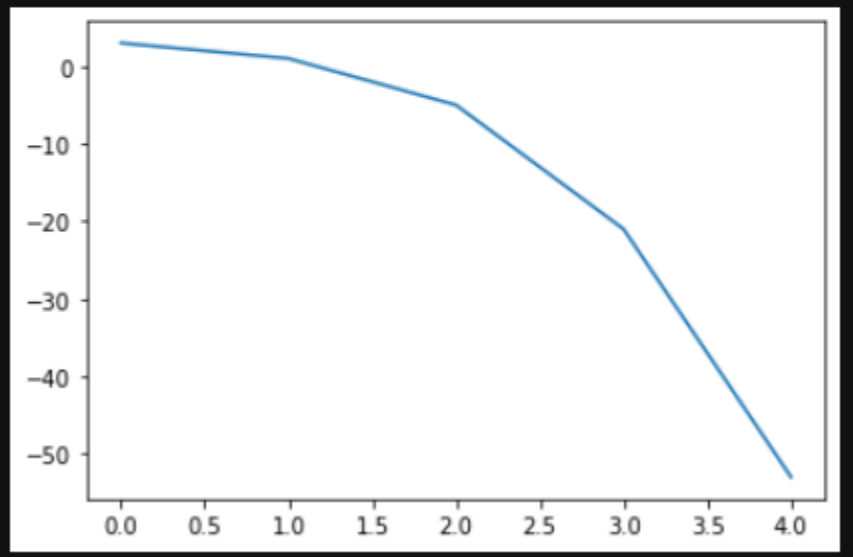
Collections : 자료형에 관한 패키지

Sklearn : 머신러닝 모델 패키지

# Polynomial Regression 예시 만들기

1. np.arange [0, 4] 까지 고르게 분할 (정수)
2. $y = 3 - 2x + x^2 - x^3$ 만들기
3. 그래프로 그리기

```
[116]: # Making polynomial regression model
       X_ex = np.arange(5)
       Y_ex =  3 - 2 * X_ex + X_ex ** 2 - X_ex ** 3

[117]: plt.plot(X_ex,Y_ex)

[117]: [<matplotlib.lines.Line2D at 0x7f6852d58278>]
```



# Polynomial Regression 예시 만들기 (실습)

1. 정의역 x_temp : 범위가 [0,1,2,3,4,5,6,7,8] 까지의 행렬을 만들기
2. 치역 y_temp = 10 * X_temp**3 + 7 * X_temp -10 만들기
3. Ploting 해보기

# Polynomial Regression 예시 만들기

1. Degree 2 polynomial feature 생성
2. $(x)$ -> $(1, x, x^2)$
3. 해당 차수에 맞게 feature 생성

```
[566]: poly = PolynomialFeatures(degree=2)
       X_ex_trans = poly.fit_transform(X_ex[:, np.newaxis])

       print("Output of Polynomial Features")
       print(X_ex_trans)
       for row in range(X_ex_trans.shape[0]):
           print("constant : {}, x1 : {}, x1**2 : {}".format(X_ex_trans[row,0],X_ex_trans[row,1]

Output of Polynomial Features
[[ 1.  0.  0.]
 [ 1.  1.  1.]
 [ 1.  2.  4.]
 [ 1.  3.  9.]
 [ 1.  4. 16.]]
constant : 1.0, x1 : 0.0, x1**2 : 0.0
constant : 1.0, x1 : 1.0, x1**2 : 1.0
constant : 1.0, x1 : 2.0, x1**2 : 4.0
constant : 1.0, x1 : 3.0, x1**2 : 9.0
constant : 1.0, x1 : 4.0, x1**2 : 16.0
```

# Polynomial Regression 예시 만들기 (실습)

```
poly_practice = PolynomialFeatures(degree=?)
X_ex_trans = poly_practice.fit_transform(X_ex[:, ??])

print("Input")
print(X_ex)
print("Output of Polynomial Features")
print(X_ex_trans)
for row in range(??):
    print("constant : {}, x1 : {}, x1**2 : {}, x1**3 : {}".format(X_ex_trans[row,0],X_ex_trans[row,1], X_ex_trans[row,2], X_ex_trans[row,3] ))
```

1. degree = 3 인 Polynomial Feature 를 poly_practice로 선언해보세요
2. input X_ex를 1에서 선언한 poly_practice를 이용하여 변환해 보세요
3. 변환 결과를 출력해보세요

# Polynomial Regression 예시 만들기

1. Pipeline :
   함수들을 엮어서 하나의 함수로 변환
2. model.fit(x, y)
   x,y 데이터를 이용하여 학습
3. model.named_steps['linear'].coef_
   Linear 모델의 계수 불러오기
4. model.predict(x)
   X를 model 에 투입하여 output 구하기

```
model = Pipeline([('poly', PolynomialFeatures(degree=2)),
                  ('linear', LinearRegression(fit_intercept=False))])
model.fit(X_ex[:, np.newaxis], Y_ex)
coefs = model.named_steps['linear'].coef_
print("Final Equation : y = {:.2f} + {:.2f}*x1 + {:.2f}*(x1**2)".format(
    coefs[0],coefs[1],coefs[2]
))
print("Original Equation y =  3 - 2 * x1 + (x1 ** 2) - (x1**3)")
```

```
Final Equation : y = 1.80 + 6.60*x1 + -5.00*(x1**2)
Original Equation y =  3 - 2 * x1 + (x1 ** 2) - (x1**3)
```

```
Y_ex_pred = model.predict(X_ex[:, np.newaxis])
plt.scatter(X_ex,Y_ex_pred,label='Predicted')
plt.plot(X_ex,Y_ex,label='Ground Truth',color='orange')
plt.legend()
```

# Polynomial Regression 예시 만들기 (실습)

```
[21]: model_practice = Pipeline([('poly', PolynomialFeatures(degree=??)),
                                 ('linear', LinearRegression(fit_intercept=False))])
      model_practice.fit(??, ??)
      coefs = ??
      print("Final Equation : y = {:.2f} + {:.2f}*x1 + {:.2f}*(x1**2) + {:.2f}*(x1**3)".format(
          coefs[0],coefs[1],coefs[2], coefs[3]
      ))
      print("Original Equation y =  3 - 2 * x1 + (x1 ** 2) - (x1**3)")

      Final Equation : y = 3.00 + -2.00*x1 + 1.00*(x1**2) + -1.00*(x1**3)
      Original Equation y =  3 - 2 * x1 + (x1 ** 2) - (x1**3)
```

1. degree = 3 인 Polynomial Regression 를 model_practice로 선언해보세요
2. X_ex, Y_ex input으로 model_practice을 학습해보세요
3. model_practice 의 계수를 출력해보세요 (상수포함 4개)
4. Ground Truth와 Predicted 값을 그래프를 그려보세요

**Polynomial 모델을 이용하여 다항 함수가 아닌 sine 함수를 근사해보도록 합시다**

1. Synthetic 데이터를 생성
2. Synthetic 데이터에 noise를 일부 추가, Train/Test set으로 분리
3. Train set을 이용하여 모델을 학습하고 Test set을 이용하여 모델의 성능을 평가
4. Regression 성능 지표인 Mean Squared loss와 Bias/Variance를 측정해봅니다

# 데이터셋 만드는 함수 정의

1. get_y_true(x)
   X에 대한 sin(x) 값 구하기
2. get_y_noise(x)
   X에 대한 noise + sin(x) 값 구하기
3. get_sample(sample_ratio, x,y)
   X, y 에서 ratio 만큼 sample 추출

[8]:
```python
np.random.seed(10)
random.seed(10)

def get_y_true(x):
    y = np.sin(x)
    return y

def get_y_noise(x):
    y = get_y_true(x) + np.random.uniform(-0.4, 0.4, len(x))
    return y

def get_sample(sample_ratio, x, y):
    m = len(x)
    m_new = int(np.round(sample_ratio*m))
    ind = random.sample(range(m), m_new)
    ind = np.sort(ind)
    x_sample = x[ind]
    y_sample = y[ind]
    y_true_sample = get_y_true(x_sample)
    return x_sample, y_sample, y_true_sample
```

# Dataset 분할하기

```
[10]: np.random.seed(10)
      rand_indicies = list(range(0,x_num))
      random.shuffle(rand_indicies)
      rand_idicies_train, rand_idicies_test =rand_indicies[:int(x_num*0.8)], rand_indicies[int(x_num*0.8):]
      x_train = x_space[rand_idicies_train]
      y_train = get_y_noise(x_train)

      x_test = x_space[rand_idicies_test]
      y_test = get_y_noise(x_test)
```

- 전체 데이터를 학습 데이터 x_train ,y_train 과 Test 데이터 x_test, y_test로 나누기

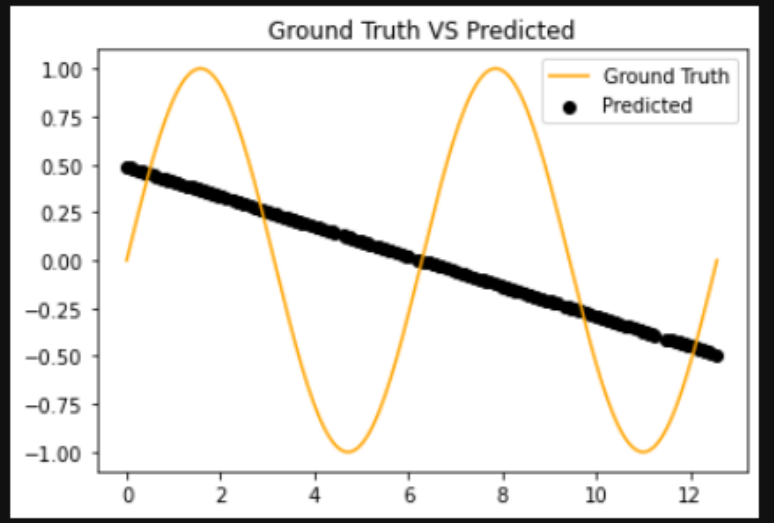
# Polynomial regression (degree =1 ) 만들기

1. degree 1 인 model 생성
2. X_train 이용하여 학습
3. 학습 결과 시각화

[13]:
```
model = Pipeline([('poly', PolynomialFeatures(degree=1)),
                  ('linear', LinearRegression(fit_intercept=False))])
model.fit(x_train[:, np.newaxis], y_train)
y_predict = model.predict(x_train[:,np.newaxis])
```

[14]:
```
plt.scatter(x_train,y_predict, label='Predicted', color ='black')
plt.plot(x_space,y, label = 'Ground Truth', color='orange')
plt.legend()
plt.title("Ground Truth VS Predicted")
```

[14]: `Text(0.5, 1.0, 'Ground Truth VS Predicted')`



# Mean Square Error 계산

1. result_dict
   결과 저장을 위한 dictionary
2. Mean_square_error 계산

$MSE = \frac{1}{n}\sum(y - \hat{y})^2$ (실제값과 예측값 차이)

[15]:
```
# Dictionary for containing the results
result_dict = defaultdict(dict)
```

**Checking Mean Square Error**

[16]:
```
y_test_predict = model.predict(x_test[:,np.newaxis])
## check Train, Test mse
mse_train = mean_squared_error(y_train,y_predict)
mse_test = mean_squared_error(y_test,y_test_predict)
result_dict[1]['mse_train'] = mse_train
result_dict[1]['mse_test'] = mse_test
print("mse Train : {}, Test : {}".format(mse_train,mse_test))
```

```
mse Train : 0.456555897331168, Test : 0.4686182390294486
```

# Bias and Variance

$$bias = \mathbb{E}[f'(x)] - f(x)$$

$$\mathbb{E}[f'(x)] = \frac{1}{n}\sum_{i=0}^{n-1} f_i'(x)$$

- 예측값의 평균과 실제값의 차이

$$variance = \mathbb{E}\left[\left(f'(x) - \mathbb{E}[f'(x)]\right)^2\right]$$

- 예측값의 분산

# Bias 구하기

```
[17]: random.seed(10)
      num_sampling = 1000
      degree = 1

      model_list = []
      exp_f_x = np.zeros(len(x_train))

      for i in range(num_sampling):
          x_train_sample, y_train_sample, y_train_true = get_sample(sample_ratio=0.3,x=x_train,y=y_train)
          model = Pipeline([('poly', PolynomialFeatures(degree=degree)),
                            ('linear', LinearRegression(fit_intercept=False))])
          model.fit(x_train_sample[:, np.newaxis], y_train_sample)
          exp_f_x += model.predict(x_train[:, np.newaxis])
          model_list.append(model)


      exp_f_x/=num_sampling
      bias = (np.linalg.norm(exp_f_x-y_train))
      print("Exp_f_X",np.mean(exp_f_x))
```

- num_sampling 만큼 sample 추출 (Dataset 만들기)
- $E[f'(x)]$ 구하기 (exp_f_x)
- Bias 계산

# Variance 구하기

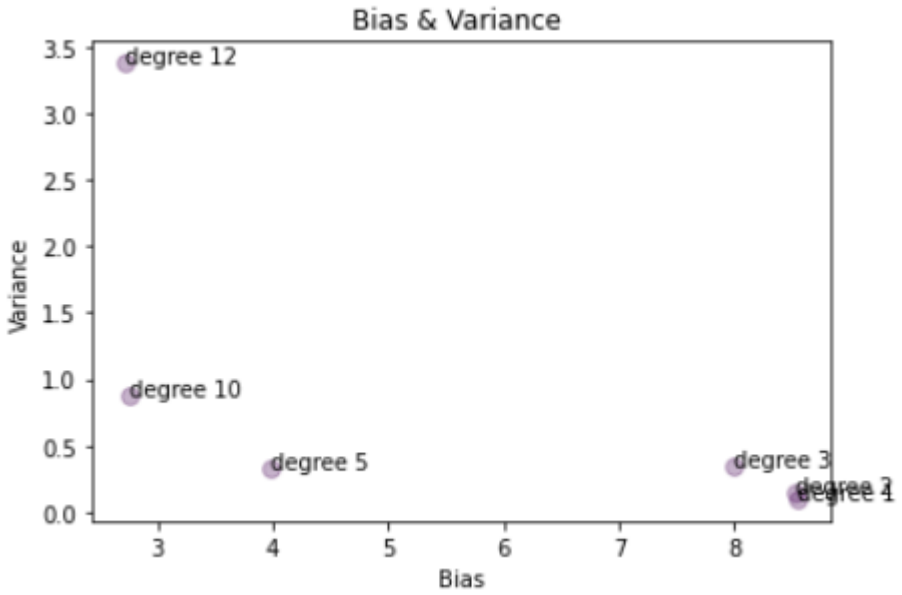
```
var = 0
for j in range(num_sampling):
    model = model_list.pop(0)
    var = var + np.square(model.predict(x_train[:, np.newaxis])-exp_f_x)
var = var/num_sampling
variance = np.linalg.norm(var)

result_dict[1]['bias'] = bias
result_dict[1]['var'] = variance
print("degree : {} bias : {} variance : {}".format(degree, bias,variance) )
```

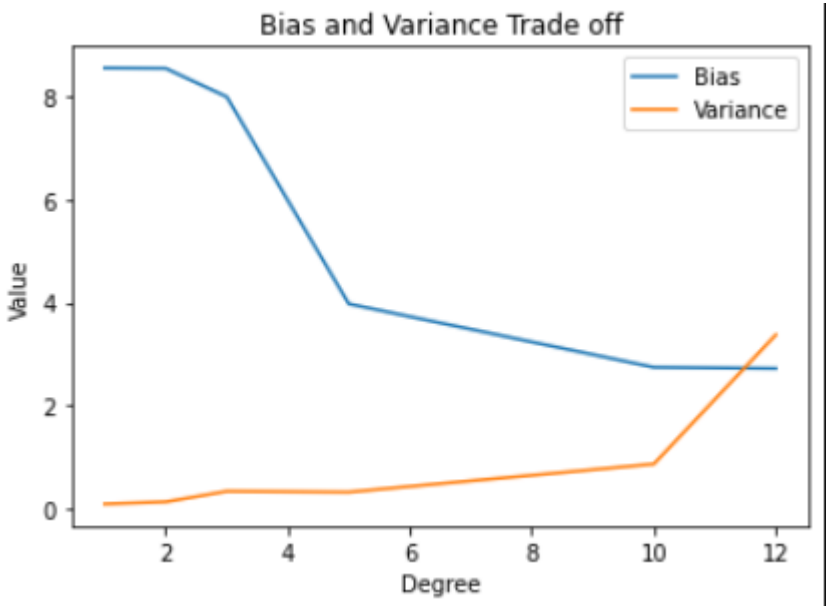
- $f'(x)$ 분산 계산

# Degree = 5 Practice

- Degree 5 일 때 직접 실습해보기
- Train MSE, Test MSE 계산
- Bias, Variance 계산

# Degree 에 따른 bias , variance 비교

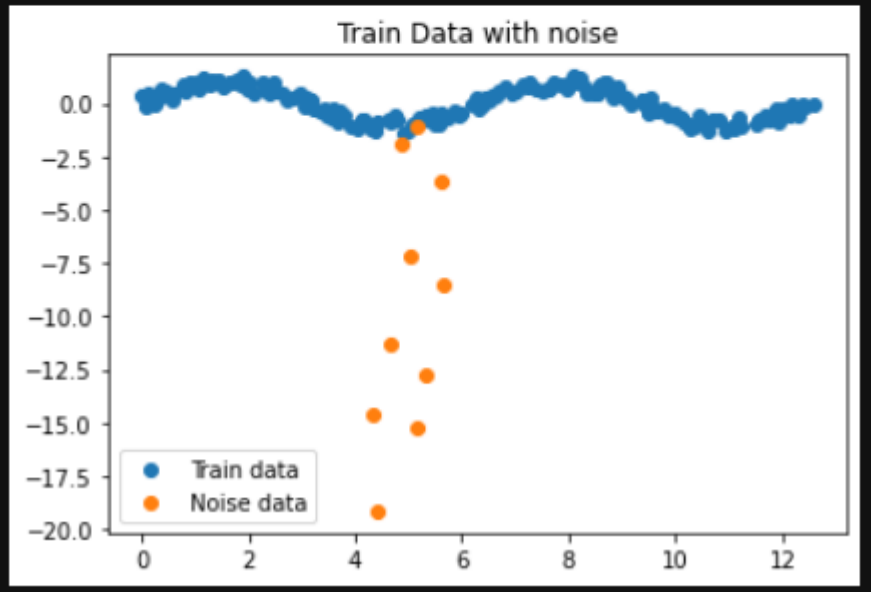


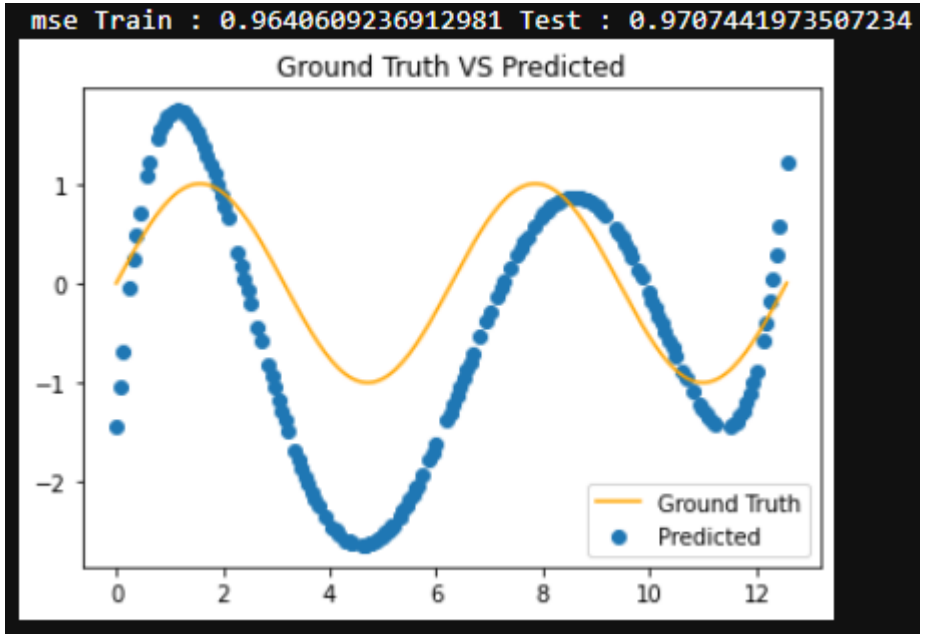
# 강한 노이즈 데이터 생성

- Random.uniform(-20,-1)
  -20에서 -1 사이의 숫자 추출 (노이즈 생성)

```
[27]: random.seed(10)
      x_noise = [random.uniform(4,6) for i in range(10)]
      y_noise = [random.uniform(-20,-1) for i in range(10)]
      plt.scatter(x_train,y_train, label='Train data')
      plt.scatter(x_noise,y_noise, label='Noise data')
      plt.legend()
      plt.title("Train Data with noise")

[27]: Text(0.5, 1.0, 'Train Data with noise')
```

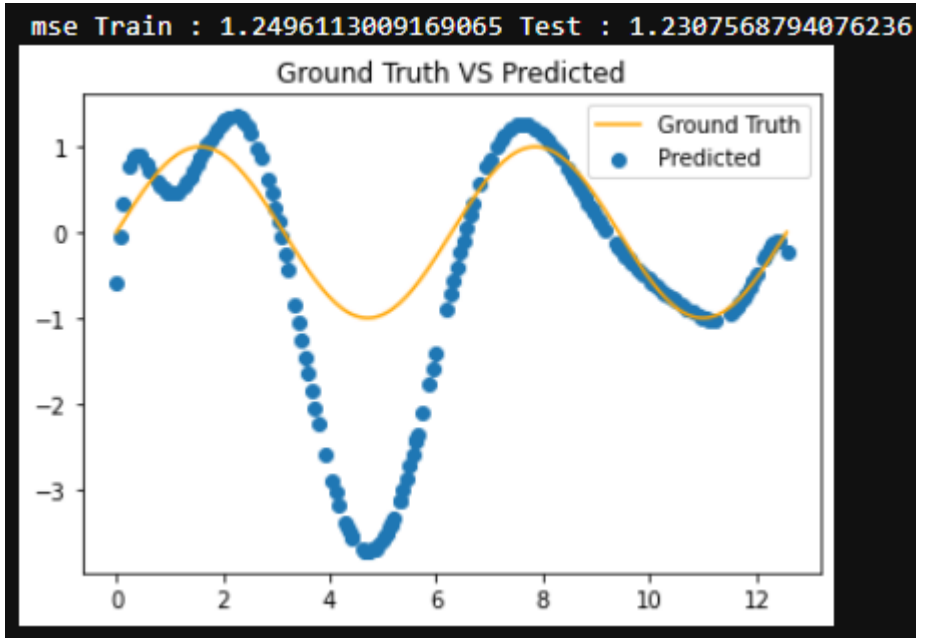


# 강한 노이즈 데이터 생성




**Degree = 5**

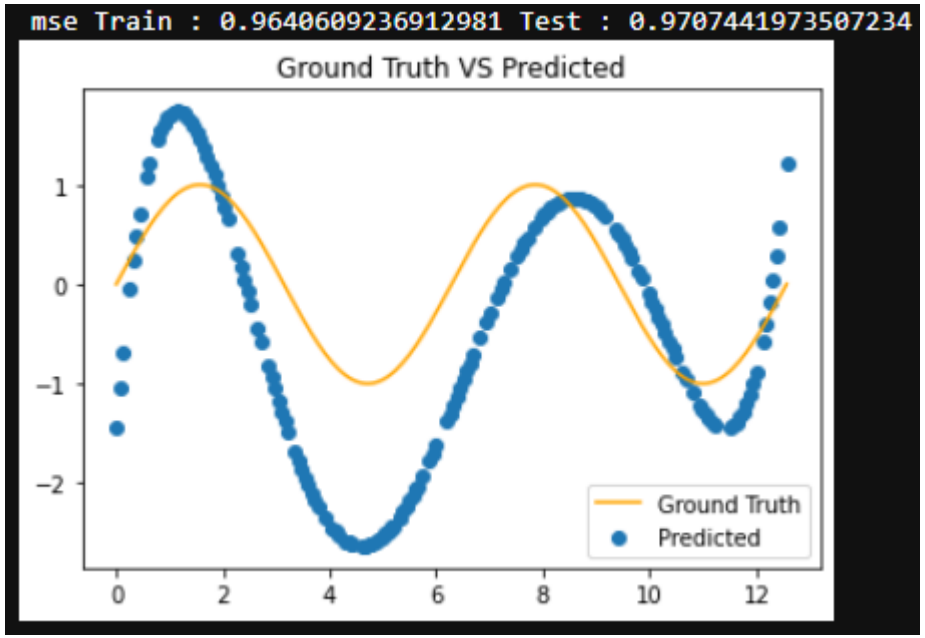
MSE_Test = 0.970




**Degree = 10**

MSE_Test = 1.207

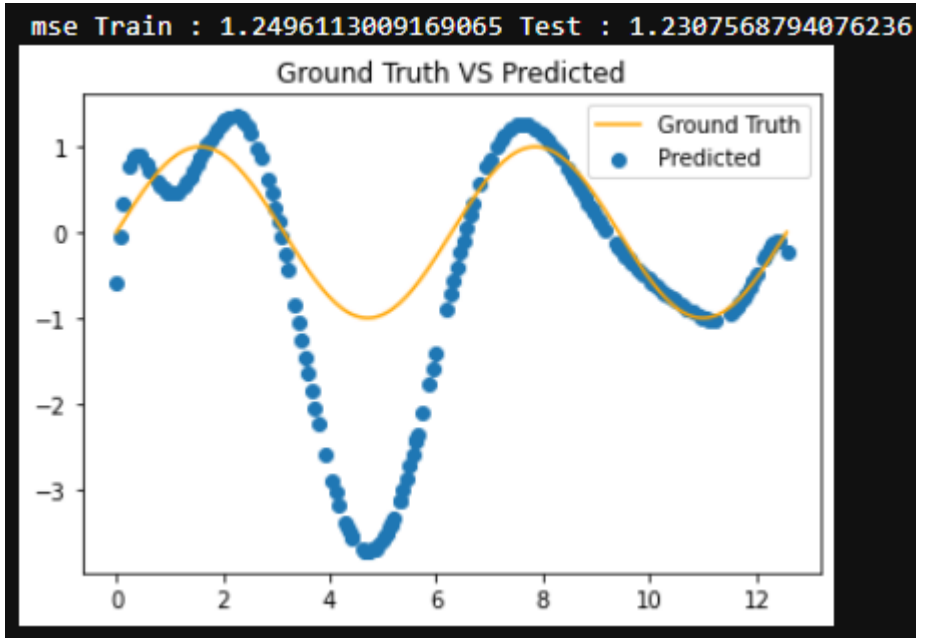
# 강한 노이즈 데이터 생성




**Degree = 5**

MSE_Test = 0.970

<




**Degree = 10**

MSE_Test = 1.207