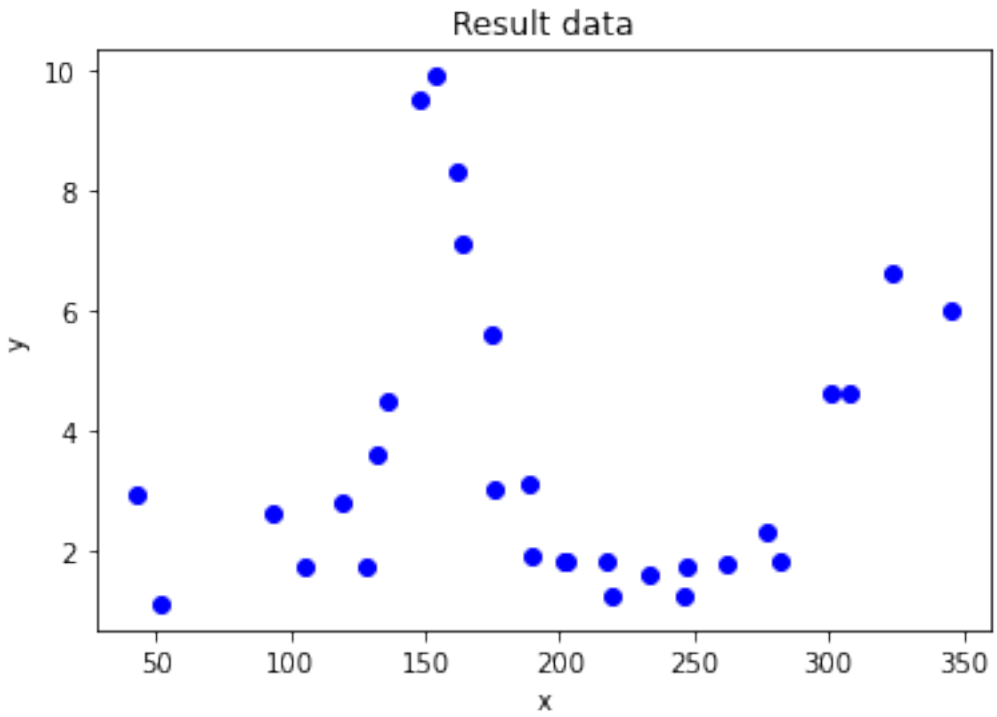
lake_mendota_clarity_starter

May 2, 2022

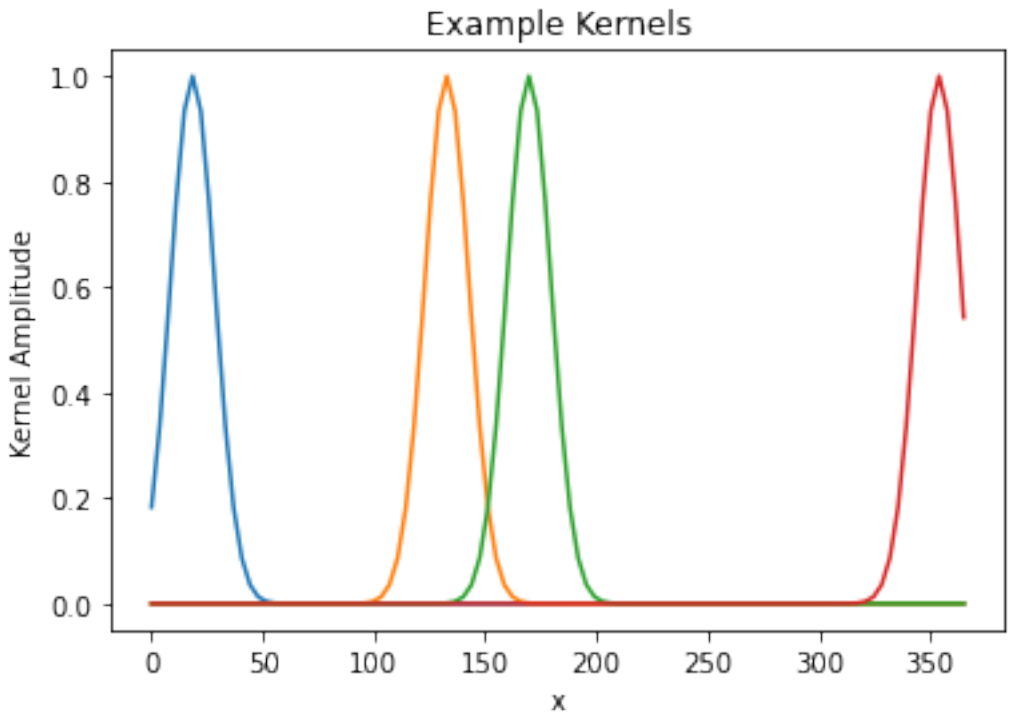
[14]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

df = pd.read_csv('mendota_secchi_depth.txt', delimiter='\t')
x = df['day_of_year']
y = df['secchi_depth']
n = len(x)
```

[15]:
```
plt.plot(x,y,'bo')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Result data')
plt.show()
```



[17]:
```
sigma = 10 #defines Gaussian kernel width
p = 100 #number of points on x-axis

# Display examples of the kernels
x_test = np.linspace(0,365,p)  # uniformly sample interval [0,365] (1 year)
j_list = [5, 36, 46, 96]  #list of indices for example kernels

Kdisplay = np.zeros((p,len(j_list)),dtype=float)

for i in range(p):
    for j in range(len(j_list)):
        Kdisplay[i,j]= np.exp(-(x_test[i]-x_test[j_list[j]])**2/(2*sigma**2))

print('Sigma = ',sigma)
plt.plot(x_test, Kdisplay)
plt.title('Example Kernels')
plt.xlabel('x')
plt.ylabel('Kernel Amplitude')
plt.show()
```

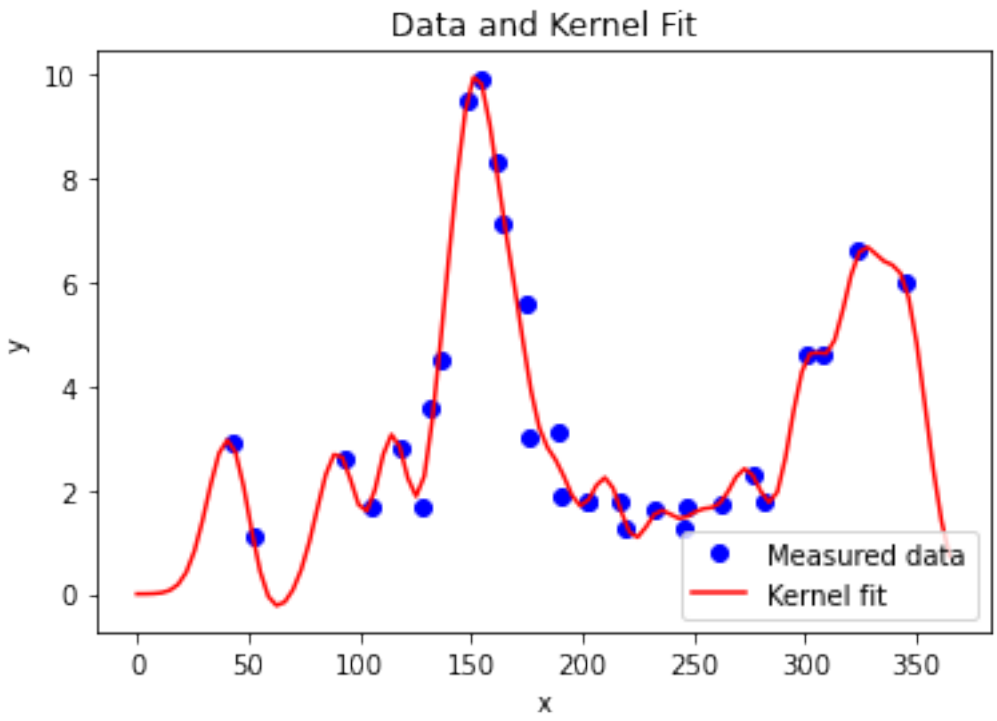
```
Sigma =  10
```



[30]:
```
# Kernel fitting to data
lam = 0.01 #ridge regression parameter

distsq=np.zeros((n,n),dtype=float)

for i in range(0,n):
    for j in range(0,n):
        distsq[i,j]=(x[i]-x[j])**2

K = np.exp(-distsq/(2*sigma**2))

alpha = np.linalg.inv(K+lam*np.identity(n))@y
```

[31]:
```
# Generate smooth curve corresponding to data fit

distsq_xtest = np.zeros((p,n),dtype=float)
for i in range(0,p):
    for j in range(0,n):
        distsq_xtest[i,j] = (x_test[i]-x[j])**2

dtest = np.exp(-distsq_xtest/(2*sigma**2))@alpha

dtrue = x_test*x_test + 0.4*np.sin(1.5*np.pi*x_test)  # noise free data for␣
 ↪comparison

print('Sigma = ',sigma)
print('Lambda = ',lam)
plt.plot(x,y,'bo',label='Measured data')
plt.plot(x_test,dtest,'r',label='Kernel fit')
# plt.plot(x_test,dtrue,'g',label='True noise free')
plt.title('Data and Kernel Fit')
plt.legend(loc='lower right')
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```

```
Sigma =  10
Lambda =  0.01
```



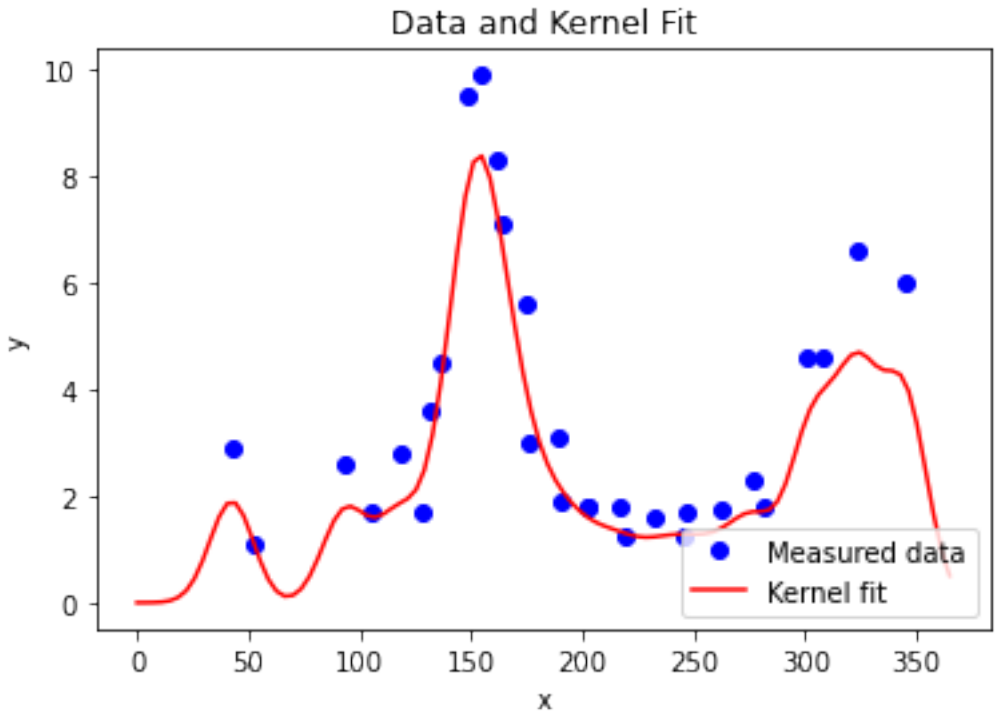
This parameters, lambda = 0.01 and sigma = 10 cleary overfit the data.

[52]:
```
# Kernel fitting to data
lam = .5 #ridge regression parameter

distsq=np.zeros((n,n),dtype=float)

for i in range(0,n):
    for j in range(0,n):
        distsq[i,j]=(x[i]-x[j])**2

K = np.exp(-distsq/(2*sigma**2))

alpha = np.linalg.inv(K+lam*np.identity(n))@y
```

[53]:
```
# Generate smooth curve corresponding to data fit

distsq_xtest = np.zeros((p,n),dtype=float)
for i in range(0,p):
    for j in range(0,n):
        distsq_xtest[i,j] = (x_test[i]-x[j])**2

dtest = np.exp(-distsq_xtest/(2*sigma**2))@alpha
```

```
dtrue = x_test*x_test + 0.4*np.sin(1.5*np.pi*x_test)  # noise free data for␣
 ↪comparison

print('Sigma = ',sigma)
print('Lambda = ',lam)
plt.plot(x,y,'bo',label='Measured data')
plt.plot(x_test,dtest,'r',label='Kernel fit')
# plt.plot(x_test,dtrue,'g',label='True noise free')
plt.title('Data and Kernel Fit')
plt.legend(loc='lower right')
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```

```
Sigma =  10
Lambda =  0.5
```



If we try a bigger lambda, as .5 we can see that the model does not overfit the data. However, because of the gaussian function we cannot find a perfect model that does not overfit nor underfit the data. The model that we want will be something like:

b) First we need to split our data into training and testing set. By doing this we then could use the testing set to test the accuracy of the model. Then we the training set into a second training set and a validation set. So we will train the model different parameters and test it with the validation

set.