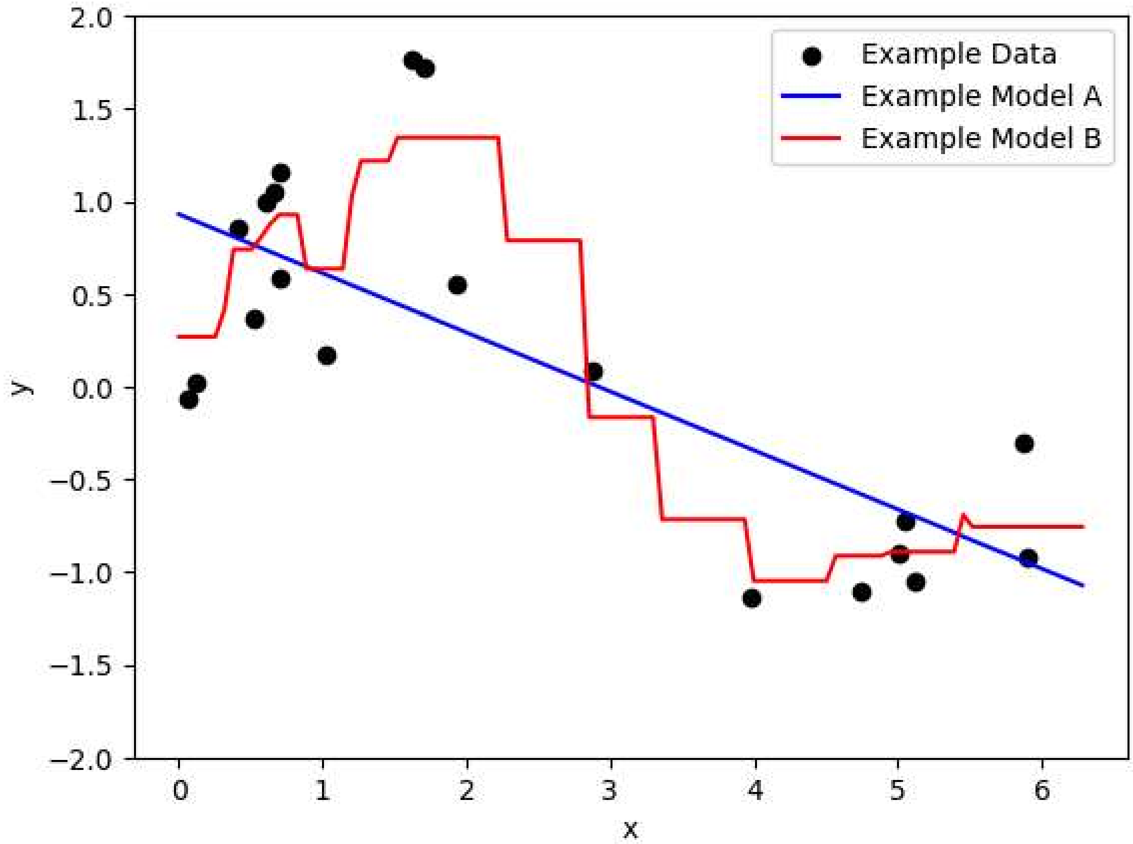
M10-L1 Problem 1

In this problem you will look compare models with lower/higher variance/bias by computing bias and variance at a single point.

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsRegressor

def plot_model(model,color="blue"):
    x = np.linspace(0, np.pi*2, 100)
    y = model.predict(x.reshape(-1,1))
    plt.plot(x, y, color=color)
    plt.xlabel("x")
    plt.ylabel("y")

def plot_data(x, y):
    plt.scatter(x,y,color="black")

def eval_model_at_point(model, x):
    return model.predict(np.array([[x]])).item()

def train_models():
    x = np.random.uniform(0,np.pi*2,20).reshape(-1,1)
    y = np.random.normal(np.sin(x),0.5).flatten()

    modelA = LinearRegression()
    modelB = KNeighborsRegressor(3)
    modelA.fit(x,y)
    modelB.fit(x,y)
    return modelA, modelB, x, y
```

The function `train_models` gets 20 new data points and trains two models on these data points. Model A is a linear regression model, while model B is a 3-nearest neighbor regressor.

In [2]:
```
modelA, modelB, x, y = train_models()
plt.figure()
plot_data(x,y)
plot_model(modelA,"blue")
plot_model(modelB,"red")
plt.legend(["Example Data", "Example Model A", "Example Model B"])
plt.ylim([-2,2])
plt.show()
```



Training models

First, train 50 instances of model A and 50 instances of model B. Store all 100 total models for use in the next few cells. Generate these models with the function: `modelA, modelB, x, y = train_models()`.

In [ ]:
```
# YOUR CODE GOES HERE
A_list = []
B_list = []
X = []
Y = []
for i in range(50):
    modelA, modelB, x, y = train_models()
    A_list.append(modelA)
    B_list.append(modelB)
    X.append(x)
    Y.append(Y)

print(A_list)
print(B_list)
```

```
[LinearRegression(), LinearRegression(), LinearRegression(), LinearRegression(), Lin
earRegression(), LinearRegression(), LinearRegression(), LinearRegression(), LinearR
egression(), LinearRegression(), LinearRegression(), LinearRegression(), LinearRegre
ssion(), LinearRegression(), LinearRegression(), LinearRegression(), LinearRegressio
n(), LinearRegression(), LinearRegression(), LinearRegression(), LinearRegression(),
LinearRegression(), LinearRegression(), LinearRegression(), LinearRegression(), Line
arRegression(), LinearRegression(), LinearRegression(), LinearRegression(), LinearRe
gression(), LinearRegression(), LinearRegression(), LinearRegression(), LinearRegres
sion(), LinearRegression(), LinearRegression(), LinearRegression(), LinearRegression
(), LinearRegression(), LinearRegression(), LinearRegression(), LinearRegression(),
LinearRegression(), LinearRegression(), LinearRegression(), LinearRegression(), Line
arRegression(), LinearRegression(), LinearRegression(), LinearRegression()]
[KNeighborsRegressor(n_neighbors=3), KNeighborsRegressor(n_neighbors=3), KNeighborsR
egressor(n_neighbors=3), KNeighborsRegressor(n_neighbors=3), KNeighborsRegressor(n_n
eighbors=3), KNeighborsRegressor(n_neighbors=3), KNeighborsRegressor(n_neighbors=3),
KNeighborsRegressor(n_neighbors=3), KNeighborsRegressor(n_neighbors=3), KNeighborsRe
gressor(n_neighbors=3), KNeighborsRegressor(n_neighbors=3), KNeighborsRegressor(n_ne
ighbors=3), KNeighborsRegressor(n_neighbors=3), KNeighborsRegressor(n_neighbors=3),
KNeighborsRegressor(n_neighbors=3), KNeighborsRegressor(n_neighbors=3), KNeighborsRe
gressor(n_neighbors=3), KNeighborsRegressor(n_neighbors=3), KNeighborsRegressor(n_ne
ighbors=3), KNeighborsRegressor(n_neighbors=3), KNeighborsRegressor(n_neighbors=3),
KNeighborsRegressor(n_neighbors=3), KNeighborsRegressor(n_neighbors=3), KNeighborsRe
gressor(n_neighbors=3), KNeighborsRegressor(n_neighbors=3), KNeighborsRegressor(n_ne
ighbors=3), KNeighborsRegressor(n_neighbors=3), KNeighborsRegressor(n_neighbors=3),
KNeighborsRegressor(n_neighbors=3), KNeighborsRegressor(n_neighbors=3), KNeighborsRe
gressor(n_neighbors=3), KNeighborsRegressor(n_neighbors=3), KNeighborsRegressor(n_ne
ighbors=3), KNeighborsRegressor(n_neighbors=3), KNeighborsRegressor(n_neighbors=3),
KNeighborsRegressor(n_neighbors=3), KNeighborsRegressor(n_neighbors=3), KNeighborsRe
gressor(n_neighbors=3), KNeighborsRegressor(n_neighbors=3), KNeighborsRegressor(n_ne
ighbors=3), KNeighborsRegressor(n_neighbors=3), KNeighborsRegressor(n_neighbors=3),
KNeighborsRegressor(n_neighbors=3), KNeighborsRegressor(n_neighbors=3), KNeighborsRe
gressor(n_neighbors=3), KNeighborsRegressor(n_neighbors=3), KNeighborsRegressor(n_ne
ighbors=3), KNeighborsRegressor(n_neighbors=3), KNeighborsRegressor(n_neighbors=3),
KNeighborsRegressor(n_neighbors=3)]
```

Bias and Variance

Now we will use the definitions of bias and variance to compute the bias and variance of each type of model. You will focus on the point x = 1.57 only. First, compute the prediction for each model at x. (You can use the function `eval_model_at_point(model, x)`).

In [14]:

```python
x = 1.57
evals_A = []
evals_B = []
# YOUR CODE GOES HERE
for model in A_list:
    eval = eval_model_at_point(model,1.57)
    evals_A.append(eval)

for model in B_list:
    eval = eval_model_at_point(model,1.57)
    evals_B.append(eval)
```

```
print(evals_A)
print(evals_B)
```

```
[0.5181475839280177, 0.773542946145354, 0.7537269250901353, 0.6026191174285678, 0.36
38258756705326, 0.37426838744644164, 1.1175399956854832, 0.5514751888012029, 0.28207
113240152526, 0.46739746608696386, 0.30104601969032974, 0.5970052693905665, 0.257229
54142666266, 0.7958836123249431, 0.29238535712134817, 0.35465824661681034, 0.3957154
037997074, 0.4431791602745607, 0.35703907934790313, 0.8257599904547256, 0.4673324861
468755, 0.9225558143831335, 0.8109245590087958, 0.6885068069332759, 0.75543609080192
56, 0.4114975302626954, 0.47235072654991384, 0.5646720904644855, 0.2498391285522124,
0.2815430872856721, 0.47100226712427207, 0.2871029401659906, 0.4989749026575878, 0.5
038711034805152, 0.3993267719742251, 0.6728035390681288, 0.4424621986381605, 0.47735
189599102096, 0.35987281661820975, 0.9792677516431763, -0.0068597715151286764, 0.667
6494969689144, 0.5322257456891035, 0.5982938364083388, 0.37473046696589696, 0.407681
5127441683, 0.43815941339544173, 0.5081668694186768, 0.5747526440699645, 0.549836088
2101819]
[1.0155555048987257, 1.2399126521795831, 0.8298435463417345, 0.7299953472912503, 0.9
527858791680298, 1.0055999836921448, 1.366120194894253, 0.9453775107570638, 0.503364
6910235788, 0.8509308888025956, 0.8004050055803722, 1.146159268391547, 0.65635126880
06755, 1.417910248353844, 0.6154140241349629, 1.0506584227191726, 0.343171154776261
2, 1.2939783604567607, 1.2827184320911003, 0.9953883926864157, 0.4620285465400067,
1.3538101796174917, 1.09663735136352, 0.8112853666447865, 1.5090287463093148, 0.6432
008994400067, 0.8979494863847309, 0.6597767350245799, 1.0845579750809904, 0.34227498
72133258, 0.6784451679844512, 0.8252422604548814, 1.3952594611562568, 0.815812568729
4101, 0.8443117489295346, 0.56888573568115, 0.6801394212379113, 0.9387610416667074,
0.5151803947597754, 1.182049633756041, 0.6357239168309614, 1.2069902997268156, 1.108
6412055936232, 1.1028183716286775, 1.1040877248160237, 0.6421861345341758, 1.0573561
012561663, 0.9366251975211736, 1.3179727905632426, 0.24985181526437442]
```

In this cell, use the values you computed above to compute and print the bias and variance of model A at the point x = 1.57. The true function value `y_GT` is given as 1 for x=1.57.

In [15]:
```
yGT = 1

# YOUR CODE GOES HERE
bias_A = np.mean(evals_A) - yGT
var_A = np.var(evals_A)

bias_B = np.mean(evals_B) - yGT
var_B = np.var(evals_B)

print(f"Model A:   Bias = {bias_A:.3f},  Variance = {var_A:.3f}")
print(f"Model B:   Bias = {bias_B:.3f},  Variance = {var_B:.3f}")
```

```
Model A:   Bias = -0.484,  Variance = 0.044
Model B:   Bias = -0.086,  Variance = 0.093
```

Questions

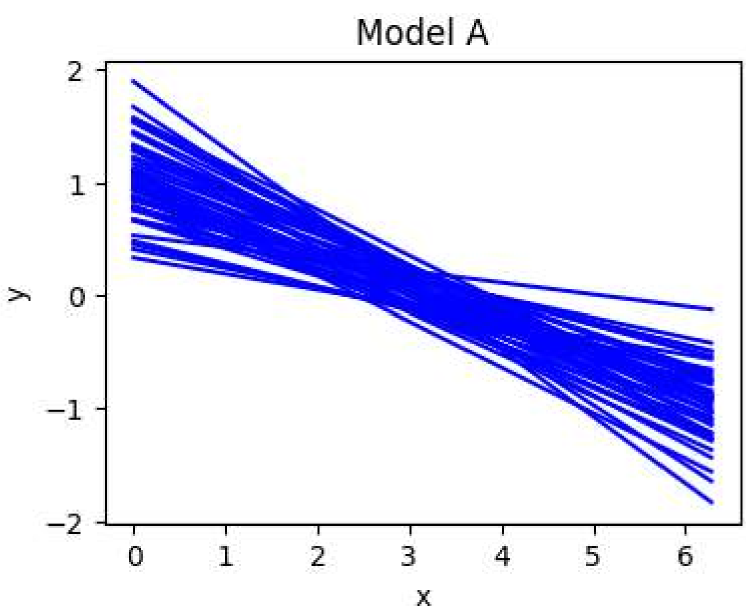
1. Which model has smaller bias at $x = 1.57$?
2. Which model has lower variance at $x = 1.57$?

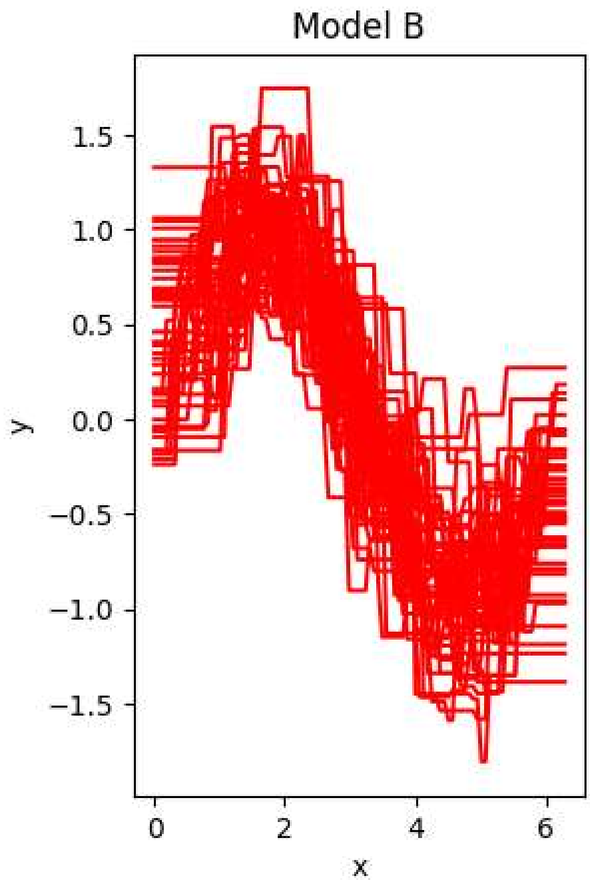
1. Model B has smaller bias at X = 1.57
2. Model A has smaller variance at x= 1.57

Plotting models

Now use the `plot_model` function to overlay all Model A predictions on one plot and all Model B predictions on another. Notice the spread of each model.

In [ ]:
```
plt.figure(figsize=(9,3))

plt.subplot(1,2,1)
plt.title("Model A")
# YOUR CODE GOES HERE
for model in A_list:
    plot_model(model, color="blue")
plt.xlabel("x")
plt.ylabel("y")
plt.show()


plt.subplot(1,2,2)
plt.title("Model B")
# YOUR CODE GOES HERE
for model in B_list:
    plot_model(model, color="red")
plt.xlabel("x")
plt.ylabel("y")
plt.show()
```





In [ ]: