CS 171 Homework 2

In this homework assignment, you will explore the application of the classification algorithms we have seen in class so far.

Import the modules for this notebook

Begin by importing any pertinent modules for classification and visualizing model results:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

from sklearn.linear_model import Perceptron, LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
# import your modules here
```

Motivation and Data

Adélie penguins are an important species on the Antarctic continent. The image below shows two Adélie penguins:

When biologists conduct fieldwork, they often need to know the sex of the penguin to understand their behavior such as foraging and mating. However, identifying the sex of a penguin either requires molecular analysis or other invasive procedures. Thus, an ideal alternative is to use a morphometric analysis - that is, collecting data on easily-measurable characteristics such as the weight or height. In this assignment, we'll build a model to classify a penguin as male or female based on measurements of body weight and two different bill metrics. The data comes from the following paper:

Fattorini, Niccolò, and Silvia Olmastroni. "Pitfalls and advances in morphometric sexing: insights from the Adélie penguin Pygoscelis adeliae." Polar Biology 44.8 (2021): 1563-1573.

The authors have kindly provided us with their data for this homework assignment and have asked us only to use this data for the purposes of this class.

To begin, let's read in the dataset below:

```
df = pd.read_csv('adelie_penguins.csv')

df.head()
```

```
  Penguin_ID  Body_weight  Bill_depth  Bill_length Sex
0       ID_1         3500       17.39        35.96   M
1       ID_2         3600       17.35        35.76   M
2       ID_3         3250       17.20        37.55   M
3       ID_4         3950       17.90        36.85   F
4       ID_5         3600       16.54        36.66   M
```

As we can see, the sex of the penguin in shown in the final column and the previous columns have other features of the penguins.

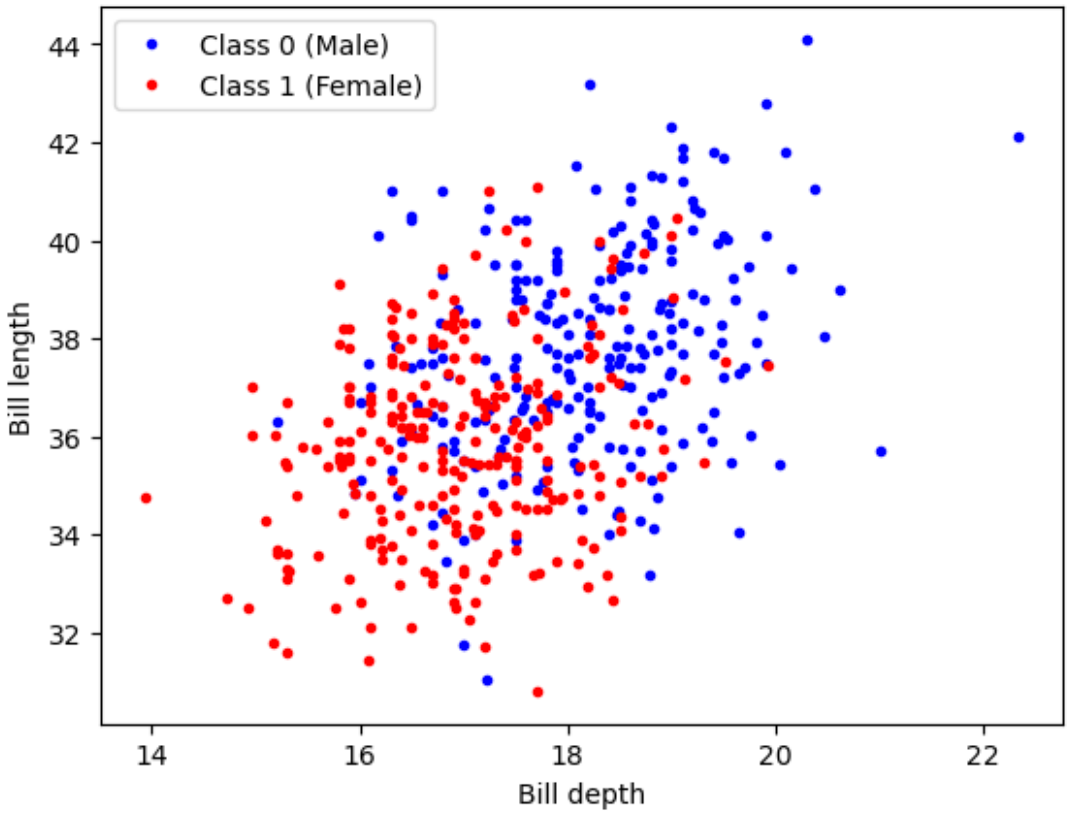
Problem 2.1: Visualize the data

To get familiar with this dataset, make a scatter plot of the data. Choose one of the penguin features to plot on the x-axis and another to plot on the y-axis. The dots should be colored by the penguin sex (male or female). Be sure to add a legend, a title, and label your axes.

```
df['Classification'] = 1
df.loc[df['Sex'] == 'M', 'Classification'] = 0
df = df.dropna()
df
```

```
    Penguin_ID  Body_weight  Bill_depth  Bill_length Sex
Classification
0         ID_1         3500       17.39        35.96   M
0
1         ID_2         3600       17.35        35.76   M
0
2         ID_3         3250       17.20        37.55   M
0
3         ID_4         3950       17.90        36.85   F
1
4         ID_5         3600       16.54        36.66   M
0
..         ...          ...         ...          ...  ..              .
..
496     ID_497         3750       15.21        36.30   M
0
497     ID_498         4100       18.19        37.01   M
0
498     ID_499         4600       18.98        37.25   M
0
499     ID_500         4800       17.73        33.21   F
1
500     ID_501         3750       16.79        39.43   F
1

[501 rows x 6 columns]
```

```
# enter code to make your plot here
plt.plot(df['Bill_depth'][df['Classification'] == 0],
df['Bill_length'][df['Classification'] == 0], 'b.', label = 'Class 0
(Male)')
plt.plot(df['Bill_depth'][df['Classification'] == 1],
df['Bill_length'][df['Classification'] == 1], 'r.', label = 'Class 1
(Female)')
plt.xlabel('Bill depth')
plt.ylabel('Bill length')
plt.legend()
plt.show()
```



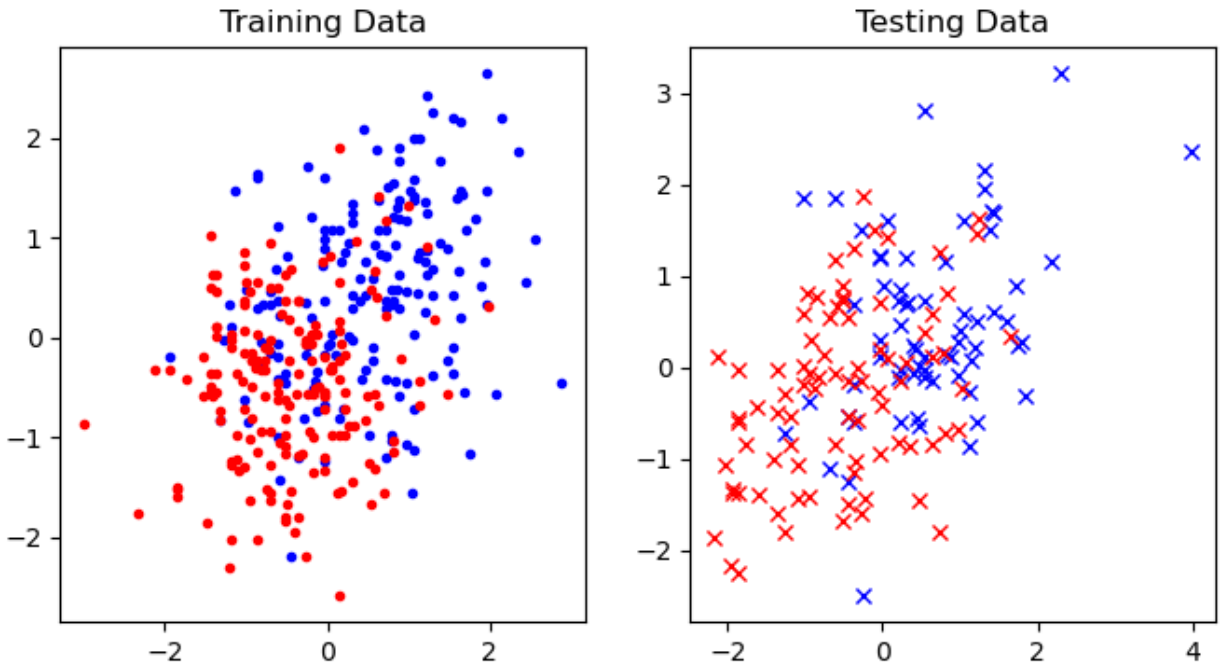
Problem 2.2: Prepare the features

In the past few lectures, we have seen several applications of classifiers. Below, you will test out two of the classifiers we have seen so far. However, when we construct our models, we need to do three organizational steps with our data:

1. Form a design matrix **X** which has the features in columns and the target features y.
2. Standardize the data
3. Split the features into a set which will be used for training and another set which will be used for validation.

Tackle these two steps in the following code blocks:

```
# form the design matrix with two of the model features
X = np.column_stack([df['Bill_depth'], df['Bill_length']])

# standardize the data
df['depth_norm'] = (df['Bill_depth'] - np.mean(df['Bill_depth'])) /
np.std(df['Bill_depth'])
df['length_norm'] = (df['Bill_length'] - np.mean(df['Bill_length'])) /
np.std(df['Bill_length'])
depth_norm = df['depth_norm']
length_norm = df['length_norm']
x = np.column_stack([depth_norm, length_norm])

# form the target data vector
y = df['Classification']

# split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size =
0.3, random_state = 17)
# plot the training and testing data
plt.figure(figsize=(8, 4))
plt.subplot(1, 2, 1)
plt.plot(X_train[y_train==0, 0], X_train[y_train==0, 1], 'b.')
plt.plot(X_train[y_train==1, 0], X_train[y_train==1, 1], 'r.')
plt.title('Training Data')

plt.subplot(1, 2, 2)
plt.plot(X_test[y_test==0, 0], X_test[y_test==0, 1], 'bx')
plt.plot(X_test[y_test==1, 0], X_test[y_test==1, 1], 'rx')
plt.title('Testing Data')
plt.show()
```



Problem 2.3: Implement a classifier

Implement one of the classifiers we have seen in class so far. Your classifier should take two of the available model features (for example, bill length and body weight) and return a classification (0 or 1) for the sex of the penguin (female or male). After fitting your model to your training data, quantify the performance by computing the loss rate (the number of correct classifications vs total classifications) for the test data.

```
# implement one of your classifiers here
model1 = LogisticRegression()
model1.fit(X_train, y_train)
```

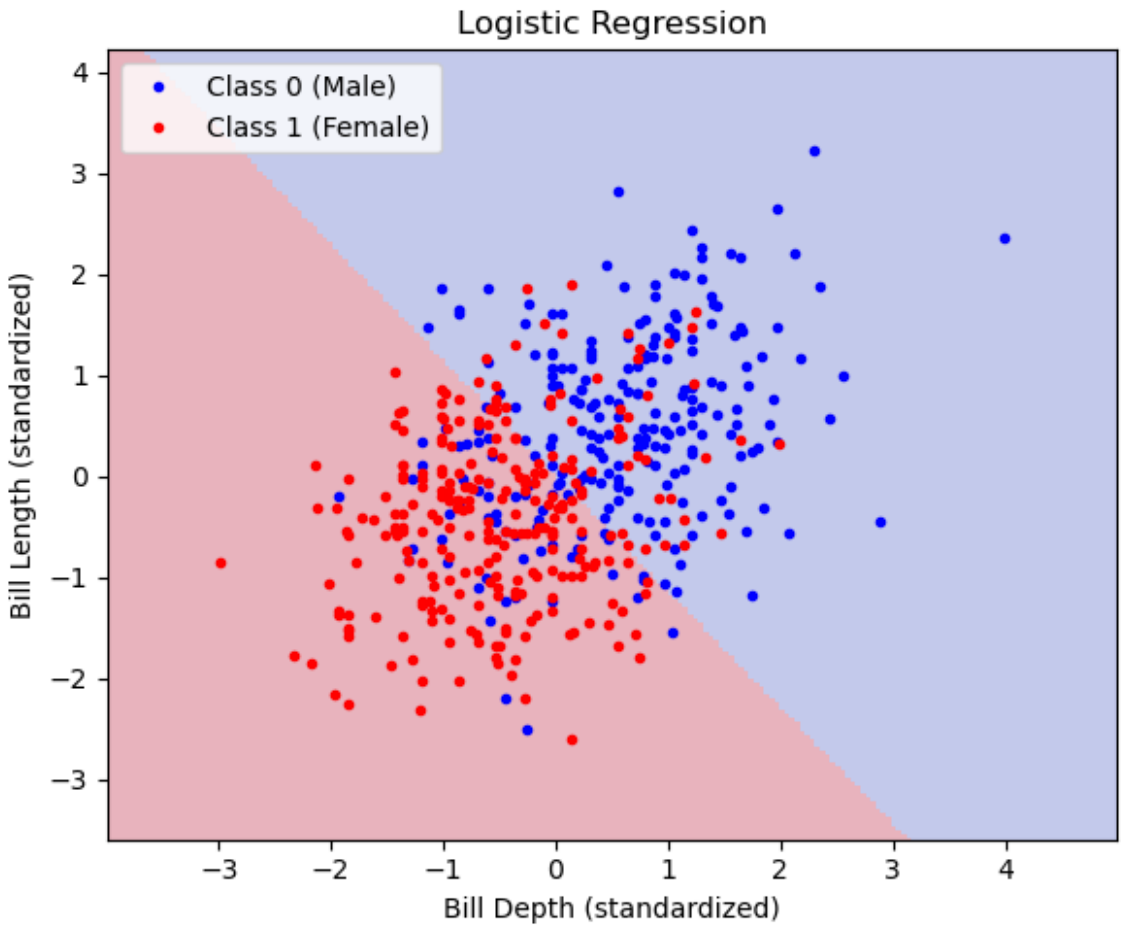
```
LogisticRegression()
```

```
# compute the loss rate predicted by your first model
y_pred = model1.predict(X_test)
train_loss = np.mean(y_pred_train != y_train)
test_loss = np.mean(y_pred != y_test)

print("weights:", model1.coef_)

print("training loss:", train_loss)
print("test loss:", test_loss)
```

```
weights: [[-1.11331803 -0.9722743 ]]
training loss: 0.24571428571428572
test loss: 0.24503311258278146
```

Visualize the performance on a plot by shading the background of the plot according to the model predictions and plotting the data on top of the shading.

```
x_min, x_max = x[:,0].min() - 1, x[:,0].max() + 1
y_min, y_max = x[:,1].min() - 1, x[:,1].max() + 1
X1, X2 = np.meshgrid(np.linspace(x_min, x_max, 200),
                     np.linspace(y_min, y_max, 200))

Y_pred = model1.predict(np.c_[X1.ravel(), X2.ravel()])
Y_pred = Y_pred.reshape(X1.shape)

fig = plt.figure(figsize=(6,5))

plt.pcolormesh(X1, X2, Y_pred, alpha=0.3, cmap='coolwarm',
shading='auto')

plt.plot(x[y==0,0], x[y==0,1], 'b.', label='Class 0 (Male)')
plt.plot(x[y==1,0], x[y==1,1], 'r.', label='Class 1 (Female)')

plt.title("Logistic Regression")
plt.gca().set_xlim([x_min, x_max])
plt.gca().set_ylim([y_min, y_max])
plt.xlabel("Bill Depth (standardized)")
plt.ylabel("Bill Length (standardized)")
plt.legend(loc=2)

plt.tight_layout()
plt.show()
```



Problem 2.4: Implement another classifier

Repeat the implementation and plotting routine above for a different classification model (but the same model features).

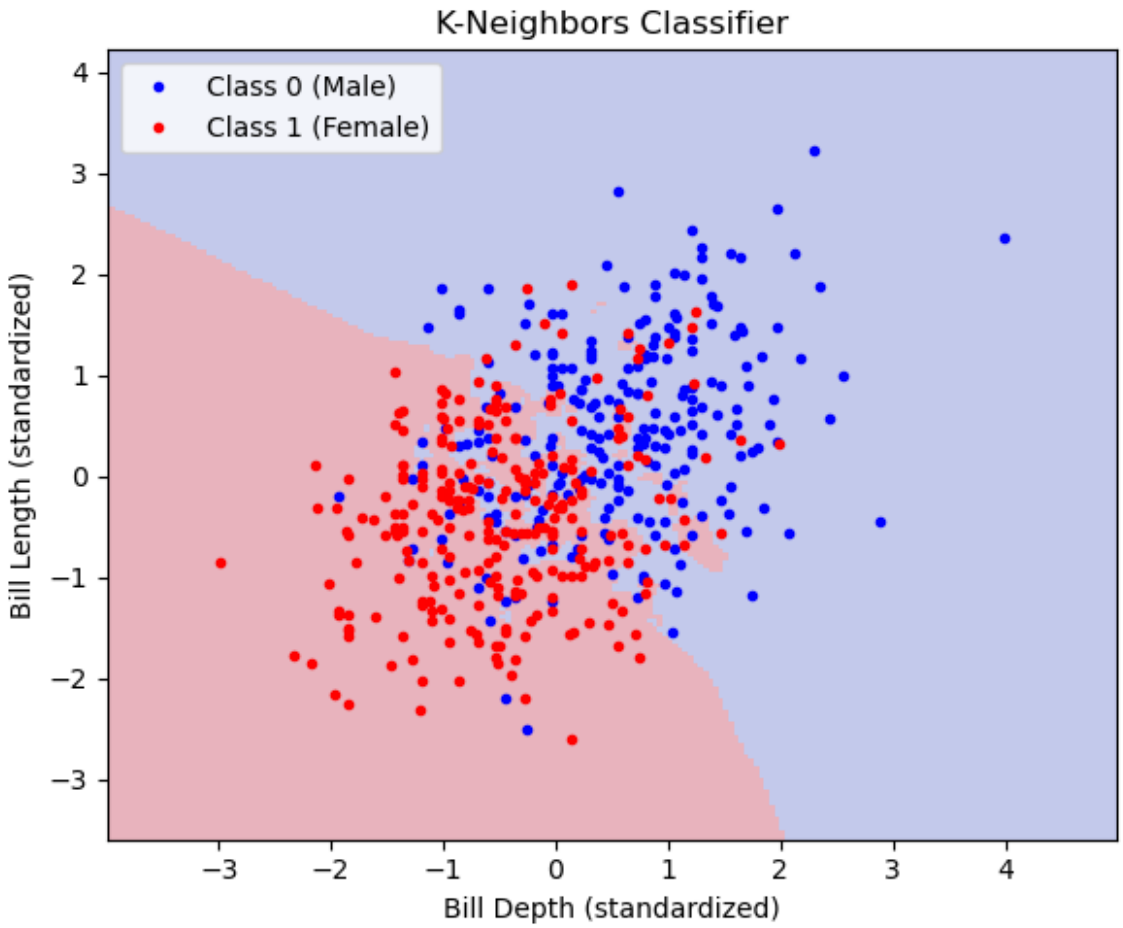
```
# implement your other classifier here
model2 = KNeighborsClassifier(n_neighbors=3)
model2.fit(X_train, y_train)
```

```
KNeighborsClassifier(n_neighbors=3)
```

```
# compute the loss rate predicted by your second model
y_pred = model2.predict(X_test)
train_loss = np.mean(y_pred_train != y_train)
test_loss = np.mean(y_pred != y_test)

print("training loss:", train_loss)
print("test loss:", test_loss)
```

```
training loss: 0.24571428571428572
test loss: 0.2582781456953642
```

```
# enter code to make your plot here
x_min, x_max = x[:,0].min() - 1, x[:,0].max() + 1
y_min, y_max = x[:,1].min() - 1, x[:,1].max() + 1
X1, X2 = np.meshgrid(np.linspace(x_min, x_max, 200),
                     np.linspace(y_min, y_max, 200))

Y_pred = model2.predict(np.c_[X1.ravel(), X2.ravel()])
Y_pred = Y_pred.reshape(X1.shape)

fig = plt.figure(figsize=(6,5))

plt.pcolormesh(X1, X2, Y_pred, alpha=0.3, cmap='coolwarm',
shading='auto')

plt.plot(x[y==0,0], x[y==0,1], 'b.', label='Class 0 (Male)')
plt.plot(x[y==1,0], x[y==1,1], 'r.', label='Class 1 (Female)')

plt.title("K-Neighbors Classifier")
plt.gca().set_xlim([x_min, x_max])
plt.gca().set_ylim([y_min, y_max])
plt.xlabel("Bill Depth (standardized)")
plt.ylabel("Bill Length (standardized)")
plt.legend(loc=2)

plt.tight_layout()
plt.show()
```



Problem 2.5: Assess model results

Write 5 sentences comparing and contrasting your model results above. Comment on the model performance on the test data, the potential for over- or under-fitting, the sensitivity to different model features, and potentially other considerations pertinent to assessing the models.

I thought both models did fairly well. The test data is very impure with most of the points overlapping so it's honestly a miracle that we got something like 0.25 loss. The logistic regression classifier is just a straight line and yet it was able to pretty much match the loss rate of the more complex K-neighbors classifier.

I think there's not much overfitting considering the loss on the training data is more or less the same as on the test data. It could be underfit judging by the gigantic loss but I think that's just a product of most of the data being really impure. Judging by the weights, the logistic regression model seems to be a little more sensitive to bill depth but I'm not sure I can say the same for the KNN model.

Problem 2.6: A little more data

A colleague has just called you from Antarctica and has asked your help in identify the sex of a few penguins in a colony they are studying. They give you the following information:

Body Weight (g)	Bill Depth (mm)	Bill Length (mm)
4000	19.04	37.81
3800	16.75	38.55
3700	18.12	39.49
4000	18.18	35.61
4500	18.62	36.87
4300	16.69	35.72

Using the above data and your models, run some computations to determine the sex of these penguins.

```
# run your computations here
data = np.array([[ 4000.0 , 19.04 , 37.81 ],
                 [ 3800.0 , 16.75 , 38.55 ],
                 [ 3700.0 , 18.12 , 39.49 ],
                 [ 4000.0 , 18.18 , 35.61 ],
                 [ 4500.0 , 18.62 , 36.87 ],
                 [ 4300.0 , 16.69 , 35.72 ]])
def probs(model):
    for i, row in enumerate(data):
        bd = row[1]
        bl = row[2]

        bd_norm = (bd - df['Bill_depth'].mean()) /
df['Bill_depth'].std()
        bl_norm = (bl - df['Bill_length'].mean()) /
df['Bill_length'].std()

        classification = model.predict([[bd_norm, bl_norm]])[0]
        probability = model.predict_proba([[bd_norm, bl_norm]])[0]

        label = 'Male' if classification == 0 else 'Female'

        print(f"Penguin {i+1}: Predicted = {label}, Probabilities =
{probability}")

probs(model1)
print()
probs(model2)
```

```
Penguin 1: Predicted = Male, Probabilities = [0.86322819 0.13677181]
Penguin 2: Predicted = Male, Probabilities = [0.51005124 0.48994876]
Penguin 3: Predicted = Male, Probabilities = [0.8462063 0.1537937]
Penguin 4: Predicted = Male, Probabilities = [0.52846824 0.47153176]
Penguin 5: Predicted = Male, Probabilities = [0.74179571 0.25820429]
Penguin 6: Predicted = Female, Probabilities = [0.2285995 0.7714005]

Penguin 1: Predicted = Male, Probabilities = [1. 0.]
```

```
Penguin 2: Predicted = Male, Probabilities = [0.66666667 0.33333333]
Penguin 3: Predicted = Male, Probabilities = [1. 0.]
Penguin 4: Predicted = Female, Probabilities = [0.33333333 0.66666667]
Penguin 5: Predicted = Male, Probabilities = [0.66666667 0.33333333]
Penguin 6: Predicted = Female, Probabilities = [0.33333333 0.66666667]
```

In the markdown cell below, write a short message to your colleague with your predictions of the penguins' sex. Be sure to include a sentence about potential errors in your approach and which penguins you may be unsure about.

I am moderately sure penguins 1, 3, and 5 are males and penguin 6 is female (~75-85% chances for these). Penguin 2 is a toss-up, both models slightly leaning on male. Penguin 4 is also a toss-up, with my models disagreeing on the gender. There is a lot of overlap in the training data so it is quite likely at least one of the "moderately sure" guesses are wrong and there's a world where every single one is wrong.