Lesson 3: From Data to Models and Predictions

Key Ideas:

1. Overview: What are mathematical models?
2. Modeling linear relationships between numerical variables (regression)
3. Using models to make predictions

1. Models

A model is a simplification of a phenomenon in the real world (an object, a process, a relationship, etc.). A model helps us understand the real-world phenomenon it represents. **A mathematical model** is a representation of a real-world object, process, or relationship in terms of mathematical objects (e.g., formulas, lines, graphs, algorithms, etc.).

A mathematical model

- is usually simplified: it is less complex than the real-world process it represents,
- involves the process of quantifying a qualitative question

George Box, a statistician, stated in his 1976 paper that "all models are wrong, but some are useful"

- Mathematical models and methods could be useful, but they are just tools and they have limitations.
- Human beings who create (or use) the models have the responsibility
  - To be transparent about the sources of data that informs the model
  - To be transparent about the assumptions/simplifications made in creating the models
  - To carefully interpret the outcomes of any quantitative models in the context of the real-world question.

Examples of mathematical models

1. A model to determine if a person is underweight, at a healthy weight, or overweight

One possible model: The Body Mass Index (BMI)

- Compute this number: $\text{BMI} = \frac{\text{mass (kg)}}{\text{height (m)}^2}$
- Based on the resulting number, determine if the person is underweight, at a healthy weight, or overweight.

This model is simplified: it ignores many other factors that influence a person's health. While the model is perhaps too simple, it could be useful, as long as the result is interpreted carefully and with the understanding that some important factors might not be taken into account.

2. A model to predict the likelihood that an offender will commit violent crimes again after release

One possible model: Correctional Offender Management Profiling for Alternative Sanctions (COMPAS).

From Wikipedia (https://en.wikipedia.org/wiki/COMPAS_(software)):

"It is a case management and decision support tool developed andowned by Northpointe (now Equivant) used by U.S. courts to assess the likelihood of a defendant becoming a recidivist."

The Violent Recidivism Risk Scale is calculated as follows:

$$s = a(-w) + a_{\text{first}}(-w) + h_{\text{violence}}w + v_{\text{edu}}w + h_{\text{nc}}w$$

where s is the violent recidivism risk score, w is a weight multiplier, a is current age, a_{first} is the age at first arrest, h_{violence} is the history of violence, v_{edu} is vocation education level, and h_{nc} is history of noncompliance. The weight, w, is "determined by the strength of the item's relationship to person offense recidivism that we observed in our study data".

Again, there are many different factors that might influence a person's probability to commit a violent crime, and this model is simplified and takes into account only a handful of them. The model have been questioned; among others, some argued that the risk computed by this model is biased against black offenders. Below is a list of some extra reading:

- Angwin, Julia, et al. "Machine Bias. ProPublica (2016)." URL: https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing (https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing) (2016).
- Response to ProPublica: Demonstrating accuracy equity and predictive parity (https://www.equivant.com/response-to-propublica-demonstrating-accuracy-equity-and-predictive-parity/)
- The Data Processing Error in a Prominent Fair Machine Learning Dataset (long version) (https://towardsdatascience.com/the-data-processing-error-in-one-of-the-most-prominent-fair-machine-learning-datasets-4fa205daa3c4)

Types of mathematical models

In the above two examples, we have mathematical formulas that **takes various inputs** (numerical quantities such as weight and height, age of first offence, or categorical quantities such as education level), and **produce a final output**:

- In the first example, the final output of the model is **a categorical value**: underweight, healthy weight, or overweight.
- In the second example, the final output of the model is **a number**: risk (or probability) of a defendant offending again upon release.

In both examples, once the model is created, they can be used to help make predictions or decisions.

In the rest of this lesson, we will consider two types of models:

- Regression: Models where the output is a numerical value
- Classification: Models where the output is a categorical value

How mathematical models are created

Models should mimic the real-world process or phenomena that it represents. To do this, we first need to identify patterns and trends in the real-world process or phenomena. That is, we first examine real data to glean patterns and insights from them.

The mathematical modeling process:

- Step 1: Find patterns in data
- Step 2: Build a model that fits the data relatively well (e.g., fit a line through the plotted data points)
- Step 3: Assess the model
- Step 4: Repeat Steps 1-3, until we have a model that represents the real world process sufficiently well
- Step 5: Use the model to make predictions/decisions

2. Modeling linear relationships between numerical variables

We can visualize the relationship between two numerical variables by creating a scatterplot, in an effort to find patterns/trends between the variables.

Example

We will use an example dataset that is provided by the seaborn package.

- The first code cell below loads the dataset as a pandas data frame which we name `geyser`. This dataset contains information about eruptions of the Old Faithful geyser at the Yellowstone National Park, and has two numerical variables: `duration` and `waiting`. The `duration` column contains the length of each eruption (in minutes) and the `waiting` column contains the time between eruptions (also in minutes)
- In the second code cell below, we create a scatterplot to help us visually see if there is a relationship between the variables `duration` and `waiting`. The scatterplot seems to indicate that there might be a positive and linear relationship between the two variables: the longer the duration of an eruption is, the longer the waiting time until the next eruption; we might be able to capture the pattern by drawing a straight line through these data points.

In [ ]:
```
import pandas as pd
import numpy as np
import seaborn as sns

# load the geyser dataset; preview it
geyser = sns.load_dataset('geyser')
geyser
```

In [ ]:
```
# create a scatter plot of duration vs. waiting (see lesson 2 for reference on the sns.relplot() funct
ion)
sns.relplot( data = geyser , x = 'duration' , y = 'waiting' )
```

Correlation Coefficient

The correlation coefficient (https://en.wikipedia.org/wiki/Pearson_correlation_coefficient) could be used to help check whether there is a strong linear relationship between two numerical variables. We can compute the correlation coefficient between pairs of numerical variables (pairs of numerical columns in a data frame) using the `.corr()` method:

```
DATAFRAMENAME.corr()
```

Example

Compute the correlation coefficient between the `duration` and the `waiting` variables in the `geyser` data frame above.

- We see that there are four numbers computed: Each number is the correlation coefficient between the variables in the corresponding rows and columns.
- The correlation coefficient between a variable and itself is always 1
- The correlation coefficient between the `duration` and the `waiting` variables is 0.9, which is very close to 1. This confirms our previous observation that there is a strong linear relationship between the two variables.

This means that a line, or an equation of the form $\text{waiting} = m \times \text{duration} + b$, is an appropriate model that captures the relationship between the two variables.

In [ ]:
```
geyser.corr()
```

Quantifying "best" fit

Given a set of data points, there are many different lines $y = mx + b$ that we can draw through them. We want to find m and b so that $y = mx + b$ is a line that ***best*** fits the data.

But we first need to quantify what we mean by "best fit". That is, we need one number that measures how far off our model (the line) is from our data (the points). In other words, we need a "metric" to measure the "distance" between the model (the line) and the data (the points). The line that minimizes this metric is the line of best fit.

Suppose that we have a candidate model (a candidate line), we can use it to make a prediction about the y value given an x value. The error of this prediction can then be measured by computing the difference between the actual y value and the y value predicted by the line. Since we have a set of multiple data points, we can summarize the error, or the distance, between the line and the data points using one of the following ways:

- Sum of the errors
- Sum of the absolute value of the errors
- Sum of the squares of the errors
- Average of the errors
- Average of the absolute value of the errors
- Average of the squares of the errors (a.k.a. The Mean Squared Error (MSE)) $\leftarrow$ the conventional choice
- etc.

Example

In the `geyser` dataset above, suppose that we propose the equation $\text{waiting} = 15 \times \text{duration} + 20$ as a model that captures the relationship between `duration` and `waiting` (perhaps we obtained this equation by eyeballing the scatterplot and sketching the line). In this model, we think of `duration` as the independent variable; knowing the duration of an eruption, we want to predict the waiting time until the next eruption.

We can compute the Mean Squared Error between this line and this set of points, as we do in the code cell below. Whatever the MSE of this particular line is, the MSE of the best-fit line should be less than or equal to it.

In [ ]:
```
# use the above equation to predict waiting time from duration:
#     predicted waiting time = 15 times duration + 20
geyser['predicted_waiting'] = 15 * geyser['duration'] + 20

# then, compute the error between the predicted waiting time and the actual waiting time of each point
geyser['error'] = geyser['waiting'] - geyser['predicted_waiting']

# compute the square of these errors:
geyser['error_squared'] = geyser['error'] ** 2

# the MSE: the average of the error_squared column:
mse = np.mean( geyser['error_squared'] )

print(mse)
```

Finding the best-fit line

The question of finding the equation of the line (or curve) that minimizes the mean-squared error is called linear regression (https://en.wikipedia.org/wiki/Linear_regression). Python has tools that lets us find the best-fit line to a set of data points.

Plotting the line of best fit

```
sns.regplot(data = DATAFRAMENAME, x = 'COLNAME1', y = 'COLNAME2' )
```

Example

Consider the `geyser` dataset again. Plot the line of best fit that models the `waiting` time as a function of `duration` .

In [ ]:
```
sns.regplot(data = geyser, x = 'duration', y = 'waiting' , ci = None )  # Use ci = None so that confid
ence intervals are not plotted
```

Finding the equation of the line of best fit

```
from sklearn.linear_model import LinearRegression
MODELNAME = LinearRegression().fit( X, Y )
```

Where

- `X` = a **dataframe** with the column of the independent variable
- `Y` = a list/array containing the values of the dependent variable

For example:

```
X = DATAFRAMENAME[[ 'INDEPENDENTVARCOLNAME' ]]
Y = DATAFRAMENAME[ 'DEPENDENTVARCOLNAME' ]
```

Then,

- `MODELNAME.coef_` : the slope of the best fit line
- `MODELNAME.intercept_` : the y-intercept of the best fit line

Example

Consider the `geyser` dataset again. Find the equation of the line of best fit that models the `waiting` time as a function of `duration` .

In [ ]:
```
# first load the LinearRegression function from the library sklearn.linear_model
#  (the sklearn library contains various modeling/machine learning tools)
from sklearn.linear_model import LinearRegression

# Specify which are the independent/predictor variables (X)
#  and which is the dependent variable (to be predicted) (Y)
X = geyser[[ 'duration' ]]  # the two pairs of square brackets is necessary here so that X is a datafr
ame, not just an array
Y = geyser[ 'waiting' ]     # Y is an array

# here, we find a line that best fits the points given by X, Y
geyser_linear_model = LinearRegression().fit( X, Y )

# extract the slope and intercept information
slope = geyser_linear_model.coef_
yintercept = geyser_linear_model.intercept_

print(slope)
print(yintercept)
```

We can verify that this line, $\text{waiting} = 10.73 \times \text{duration} + 33.47$, has a smaller MSE than the line that we "eyeballed" in the previous example:

In [ ]:
```
geyser['predicted_waiting_bestfit'] = slope[0] * geyser['duration'] + yintercept

# then, compute the error between the predicted waiting time and the actual waiting time of each point
geyser['error_bestfit'] = geyser['waiting'] - geyser['predicted_waiting_bestfit']

# compute the square of these errors:
geyser['error_bestfit_squared'] = geyser['error_bestfit'] ** 2

# the MSE: the average of the error_squared column:
mse_bestfit = np.mean( geyser['error_bestfit_squared'] )

print(mse_bestfit)
```

Even though we haven't really proved that the best fit line indeed have the smallest possible MSE, we verify that the MSE of this best fit line is smaller than the MSE of the line that we proposed based on a visual observation.

3. Using models to make predictions

Once we have a model, we can use it to make predictions.

Example

Using the `geyser` dataset above, we created a model that relates the duration of an eruption to the waiting time until the next eruption:

$$\text{waiting} = 10.73 \times \text{duration} + 33.47$$

Suppose that we have just observed an eruption that lasted for 4.2 minutes. We can then use the model to predict the amount of time until the next eruption.

In [ ]:
```
dur = 4.2
wait_predicted = slope * dur + yintercept

print( wait_predicted )
```

In [ ]:
```

```

Lesson 3 Exercises

Exercise 1 *(correlation coefficient, linear regression)*

In the code cell below, we revisit the dataset of top 50 songs in Spotify in 2020, and store them as a data frame called `topsongs`.

1. Determine whether there is a relatively strong linear association between the energy and loudness variables in this dataset. (Hint: you could do this visually and by computing a number that quantifies the strength of a linear relationship.)
2. Whether or not you conclude that a line is an appropriate model for capturing the relationship between energy and loudness, we can construct a linear regression line relating the two variables.
   - Please find the equation for the line of best fit, with loudness as the independent variable and energy as the dependent variable.
   - Please plot this line of best fit.
   - Given on this dataset and the best fit line, please compute the Mean Square Error.
3. Based on the line of best fit you found in Question 2 above: Suppose that a new song has an energy value of 0.91. What would you predict its loudness value to be?

In [ ]:
```
import pandas as ... # import the necessary libraries
import ... as ...

topsongs = pd.read_csv('spotifytoptracks.csv')

# your work for exercise 1 here
# ...
```

In [ ]:
```
# ...
```

...