261072449_Assignment_4

March 15, 2024

NL2DS - Winter 2024

Assignment 4 – Psycholinguistic data, sound symbolism, regression, classification

Name: Sam Zhang

Student ID: 261072449

1 Instructions

This is a long homework, consisting of 72 points + 9 extra credit points. Different problems/questions will be easier for students with more programming versus more linguistics experience.

The homework will be graded out of 56 points. Thus, you do not need to actually complete the whole homework to get full credit, and are welcome to skip problems/questions. (However, note that some problems/questions require answering certain earlier problems/questions.)

There are two types of exercise:

- "Problems" require writing code.
  - Replace `# Put your answer here` with your answer.
  - The code block should run when all code above it in this file has also been run.
  - If you skip some problems, it's your responsibility to make sure that all code blocks which you filled out still run.
- "Questions" require writing text. Replace "**put your answer here**" with your answer.

For "Problems": * **Most code you'll need to complete the problems (about 80%) involves copying and modifying code from the CoLab notebooks on Regression, Classification, and Tree Methods.**
* Make sure you are very familiar with these notebooks and the code they contain. * Every `# Put your answer here` can be solved by a few lines of code, often 1-2 lines.
* **Do not reimplement any major functionality, such as train/test splits, calculating R^2, etc.** * Following the contents of these CoLab notebooks, you should: * Use `sklearn` functionality as much as possible for machine learning tools. (For example, do not fit a linear regression in Part 1 problems using another Python package.) * Use `pandas` functionalty as much as possible for basic data manipulation and analysis. * For all commands that involve randomness (fitting a regression, doing a train/test split, etc.), **please use `random_state=42` as an argument**. You will not lose points for not doing this, but using a fixed random seed will make your assignment

easier to grade. * Do not delete any code. Only add code by replacing `# Put your answer here`. This is important for grading.

Please make sure to follow directions carefully, including maximum lengths for "Question" answers. Failure to follow directions may result in partial or no credit for the relevant problem/question.

2 Part 1: Regression with psycholinguistic data

The first part of this problem set will examine some *lexical decision* data. You can read about lexical decision experiments in the wikipedia article here. (The dataset also contains so-called *speeded naming* data. You can read about that in the speeded naming section of the first paper.)

The collection of the lexical decision data is originally described in:

Balota, D. A., Cortese, M. J., Sergent-Marshall, S. D., Spieler, D. H., and Yap, M. J. (2004). Visual word recognition of single-syllable words. *Journal of Experimental Psychology: General*, 133(2):283–316.

In the following paper, this data was reanalyzed using some new features (predictors).

R. H. Baayen, L. Feldman, and R. Schreuder. Morphological Influences on the Recognition of Monosyllabic Monomorphemic Words. *Journal of Memory and Language*, 53:496– 512, 2006.

This data is discussed in Harald Baayen's book on linguistic data analysis.

Baayen, R. H. (2008). Analyzing Linguistic Data: A practical introduction to statistics. Cambridge University Press.

Our data file, `english_a4.csv`, was derived from the original data available as as the `english` dataframe of the languageR package.

Copy the data to your Drive folder from here.

[6]:
```
# throws an error if your Drive folder doesn't contain english_a4.csv
from google.colab import drive
drive.mount('/content/drive/')
!ls "/content/drive/My Drive/english_a4.csv"
```

```
Drive already mounted at /content/drive/; to attempt to forcibly remount, call
drive.mount("/content/drive/", force_remount=True).
'/content/drive/My Drive/english_a4.csv'
```

[7]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# set the random state
RANDOM_STATE = 42
np.random.seed(RANDOM_STATE)
```

2.1 Problem 1 (2 points)

Use Pandas to:

- Read the CSV file into a DataFrame called `english`.
- "Display" the dataset, similarly to how we've examined datasets in CoLab notebooks. The command you use should print the number of rows and columns at the end.

[8]:
```python
english = pd.read_csv("drive/My Drive/english_a4.csv")
display(english)
```

```
      RTlexdec  RTnaming    Word  Familiarity AgeSubject WordCategory  \
0     6.543754  6.145044     doe         2.37      young            N
1     6.304942  6.143756  stress         5.60      young            N
2     6.424221  6.131878    pork         3.87      young            N
3     6.450597  6.198479    plug         3.93      young            N
4     6.531970  6.167726    prop         3.27      young            N
…        …       …     …          …        …          …
4561  6.753998  6.446513     jag         2.40        old            V
4562  6.711022  6.506979    hash         3.17        old            V
4563  6.592332  6.386879    dash         3.87        old            V
4564  6.565561  6.519884   flirt         4.97        old            V
4565  6.667300  6.496624    hawk         3.03        old            V

      WrittenFrequency  WrittenSpokenFrequencyRatio  FamilySize  \
0             3.912023                     1.021651    1.386294
1             6.505784                     2.089356    1.609438
2             5.017280                    -0.526334    1.945910
3             4.890349                    -1.044545    2.197225
4             4.770685                     0.924801    1.386294
…                    …                          …          …
4561          2.079442                    -1.686399    1.386294
4562          3.663562                     0.436718    1.609438
4563          5.043425                     0.504395    1.945910
4564          3.135494                     0.062801    1.945910
4565          4.276666                     1.049822    1.945910

      DerivationalEntropy  …    ConfbN  NounFrequency  VerbFrequency  CV  \
0                 0.14144  …  8.833900             49              0   C
1                 0.06197  …  5.817111            565            473   C
2                 0.43035  …  2.564949            150              0   C
3                 0.35920  …  0.000000            170            120   C
4                 0.06268  …  2.197225            125            280   C
…                   …  …       …            …            …  ..
4561              0.30954  …  0.000000             10              7   C
4562              0.15110  …  0.693147             38              7   C
4563              0.63316  …  0.693147            113            231   C
4564              0.99953  …  4.304065             10             66   C
4565              0.95422  …  5.552960            109             47   C
```

```
      Obstruent  Frication      Voice  FrequencyInitialDiphoneWord  \
0          obst      burst     voiced                    10.129308
1          obst  frication  voiceless                    12.422026
2          obst      burst  voiceless                    10.048151
3          obst      burst  voiceless                    11.796336
4          obst      burst  voiceless                    11.991567
…           …         …         …                           …
4561       obst  frication     voiced                     8.311644
4562       obst  frication  voiceless                    12.567203
4563       obst      burst     voiced                     8.920923
4564       obst  frication  voiceless                    10.425639
4565       obst  frication  voiceless                     9.054388

      FrequencyInitialDiphoneSyllable  CorrectLexdec
0                           10.409763             27
1                           13.127395             30
2                           11.003649             30
3                           12.163092             26
4                           12.436772             28
…                                 …             …
4561                         8.390041             29
4562                        12.665546             29
4563                         9.287764             29
4564                        10.932142             29
4565                         9.148252             30

[4566 rows x 36 columns]
```

2.2 Question 1 (3 points)

You'll first familiarize yourself with the dataset by briefly examining the two papers above.

First, read the Wikipedia article on lexical decision, and briefly explain the lexical decision experimental task. Your answer should address: why do experimenters use this task, what is being measured, and how are conclusions reached on the basis of the results?

Q1: put your answer here (3 sentences max)

Now let's turn to the two research papers: Balota et al. (2004) and Baayen et al. (2006).

Start with the earlier paper then move on to the later paper. Note these two papers are long and use a lot of technical jargon from the field of psycholinguistics. *Reading each paper carefully would take several hours and you probably would not be able to understand everything unless you have previous familiarity with experimental psychology.* This is not the goal of this part of the assignment. The goal is to just familiarize yourself as efficiently as possible with what some of the columns in the data set mean. An important skill in data science is quickly evaluating the high level idea and questions studied in a paper and finding the places where quantitites are defined, without doing a careful reading.

A good way to approach this is to first read the abstract, the introduction and the conclusion and then have a look at the figures, always keeping in mind the data from the CSV above and trying to find interpretations for the various columns. Don't get stuck on stuff you don't understand unless you are pretty sure you need to understand it to answer the question.

Focus on figuring out where you can find the relevant information to answer the following questions.

2.3 Question 2 (2 points)

In these studies, using this dataset, various regression models are used to analyze the experimental data. What variable or variables were measured in these studies that corresponds to $\mathbf{y}$ in our notation from class (i.e., the quantities to be predicted) and which column or columns in the dataset have these values?

Answer: In the paper by Visual Word Recognition of Single-Syllable Words by Balota et al. in 2004, the predicted quantities are the reaction times in lexical decision (`RTlexdec`) and the reaction time in naming (`RTnaming`). These respectively refer to the duration between the presentation of the stimulus and the participant's response, and the duration between the participant being presented a word and their verbal response.

2.4 Question 3 (4 points)

In both papers a number of different quantities are used as predictors (or "features") for the experimental measures. These correspond to the columns of our $\mathbf{X}$ matrix from class, e.g. when we considered linear regression.

Note that between these two papers there are a lot of variables, and this a lot of columns in the table. Please determine the meaning of the following features: `Familiarity`, `AgeSubject`, `WordCategory`, `WrittenFrequency`, `WrittenSpokenFrequencyRatio`, `FamilySize`, `InflectionalEntropy`, `LengthInLetters`, `Voice`.
You will be graded on a random subset of your descriptions (about half).

Answer: - Familiarity: A subjective measure of how frequently a word feels encountered by individuals, not directly mentioned but inferred from the discussion on frequency and subjective norms.

- AgeSubject: Categorical binary feature that states whether the participant is young (mean age 20.5 years, SD 2.0) or old (mean age 73.6 years, SD 5.1)
- WordCategory: Classification of words based on their syntactic roles, such as nouns or verbs.
- WrittenFrequency: The number of times a word appears in written texts, affecting recognition and processing speeds.
- WrittenSpokenFrequencyRatio: Compares the frequency of words in written versus spoken form, indicating differences in usage across mediums.
- FamilySize: The number of related words sharing the same root.
- InflectionalEntropy: A measure of the diversity within a word's inflectional forms, reflecting morphological complexity. LengthInLetters: The number of letters in a word, influencing recognition and processing times.
- Voice: If the word was presented visually or orally.

For each of these predictors, think about: how would you intuitively expect it to relate to the reactions times in the **y** variables? **This is not a graded question**, but it is referred to below.

(optional) put your answer here

2.5 Problem 2 (3 points)

The largest effect in this data is age: younger participants have lower reaction times. Some predictors' effects may in fact differ between younger and older participants. To abstract away from this for this assignment, we will restrict to just data from younger participants.

We will also abstract away from the fact that a couple of the predictors here, `WordCategory` and `Voice`, are categorical. Instead we'll code them as 0/1 valued, so that:

- `WordCategory` = *N* / *V* becomes 0/1
- `Voice` = *voice* / *voiceless* becomes 0/1

Let's simplify the dataset as follows, saving to a new dataframe called `english_young`:

- Drop rows which don't correspond to young speakers, then drop the column indexing whether speakers are old or young.
- Keep the column for lexical decision RT, which will be our **y**, and drop any other columns that are possible outcome variables (from your answer to Question 2).
- Keep the column for `Word`, which tells us what word (of English) each row corresponds to.
- Recode the `WordCategory` and `Voice` columns as numeric, as specified above.
- Keep columns corresponding to the remaining predictors from Question 3.
- Drop all other columns.

Then, print a one-line message giving the number of rows and columns in `english_young`.

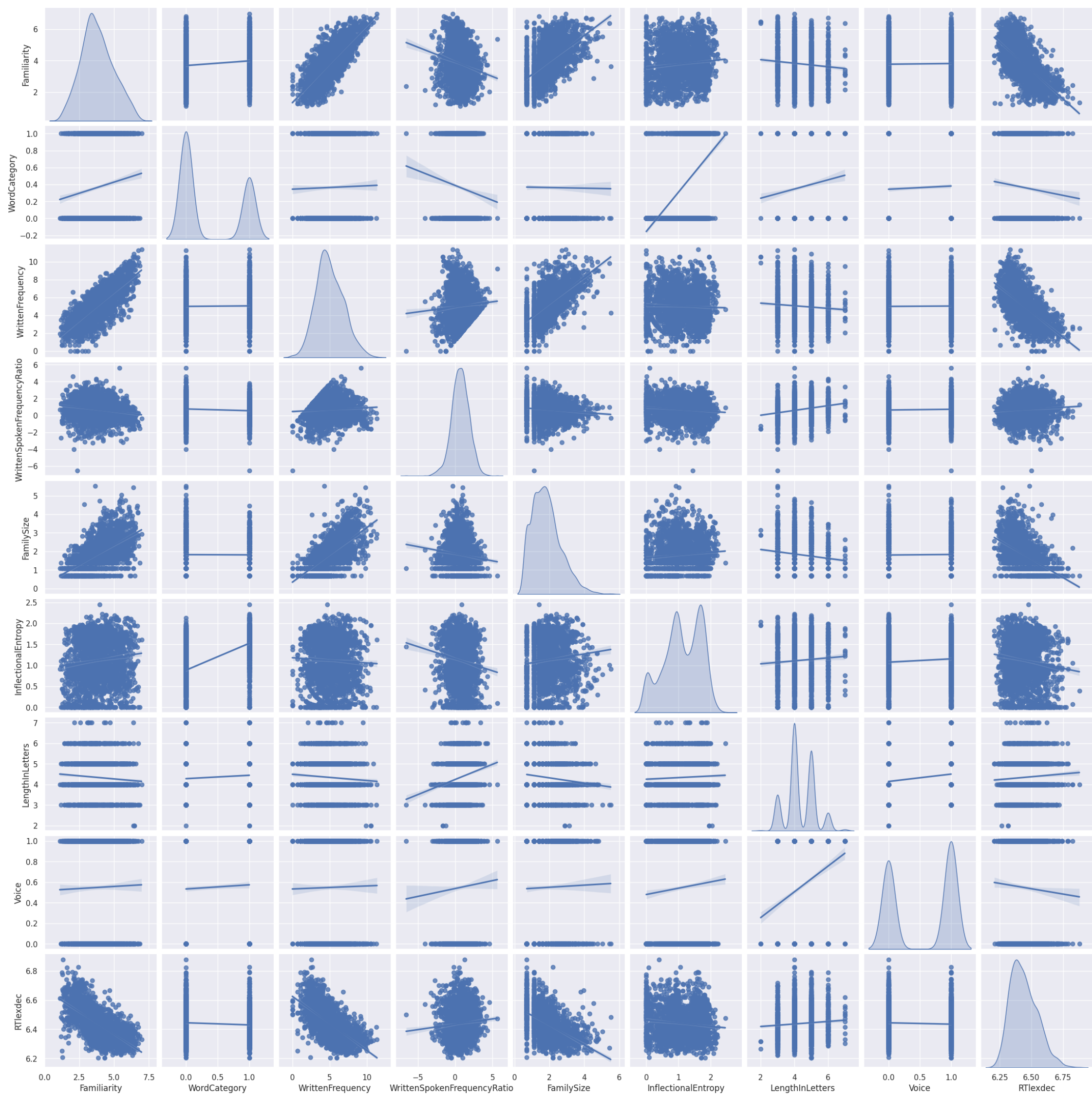
[9]:
```
# Problem 2:

# subset to young speakers
english_young = english[english['AgeSubject'] == 'young'].
 ↪drop(columns=['RTnaming'])

## map categorical predictors to numeric
category_to_number_dict = {'N': 0,'V': 1, 'voiced': 0,'voiceless': 1}

english_young['WordCategory'] = english_young['WordCategory'].
 ↪map(category_to_number_dict)
english_young['Voice'] = english_young['Voice'].map(category_to_number_dict)
english_young = english_young.filter(['Familiarity', 'AgeSubject',␣
 ↪'WordCategory', 'WrittenFrequency', 'WrittenSpokenFrequencyRatio',␣
 ↪'FamilySize', 'InflectionalEntropy', 'LengthInLetters', 'Voice', 'Word',␣
 ↪'RTlexdec'])

print(f"{english_young.shape[0]} rows, {english_young.shape[1]} columns")
```

```
2283 rows, 11 columns
```

We now use the Seaborn library to produce a set of plots between (see `pairplot`) all the variables in the dataset:

[10]:
```
import seaborn as sns; sns.set()
## kind = 'reg': add linear trend lines
## diag_kind = 'kde' : show density plots for each predictor in diagonal panels.
sns.pairplot(english_young, kind = 'reg', diag_kind='kde')
```

[10]:
```
<seaborn.axisgrid.PairGrid at 0x7fd504828eb0>
```



Do the panels in the first row show the patterns you predicted in the ungraded question above (positive vs. negative slope of the line)?

2.6 Problem 3 (4 points)

Let's examine the relationship between the written frequency of a word and its lexical decision time.

When exmaining relationships between two variables, especially when we're not sure if they're linear, it's useful to look at a *locally-smoothed regression line* that relates the x and y axes of a plot. This is a kind of regression model where the function is refit localy for many subsets of the data then a smooth line is interpolated between these points. One standard technique for this is known as *locally weighted scatterplot smoothing* or LOWESS.

When examining large datasets like this one, it's important to format how the data is displayed so that both the empirical distribution of data and the fitted trend (here, linear or LOWESS line) are legible, meaning: * Points should not overlap too much * Neither points nor the trend is formatted such that the other is obscured.

Other desiderata for any plot are: * x and y axes should be clearly labeled (with interpretable labels, not variable names like `RTlexdec`) * Text should be legible: appropriately-sized fonts, no overlapping text.

Use functions from matplotlib and seaborn to make **legible** plots meeting the specifications above:

- Make a 1 x 2 grid of plots
- In the left plot, put a scatterplot of written frequency (x-axis) and lexical decision RT (y-axis), with a superimposed linear trend (line of best fit).

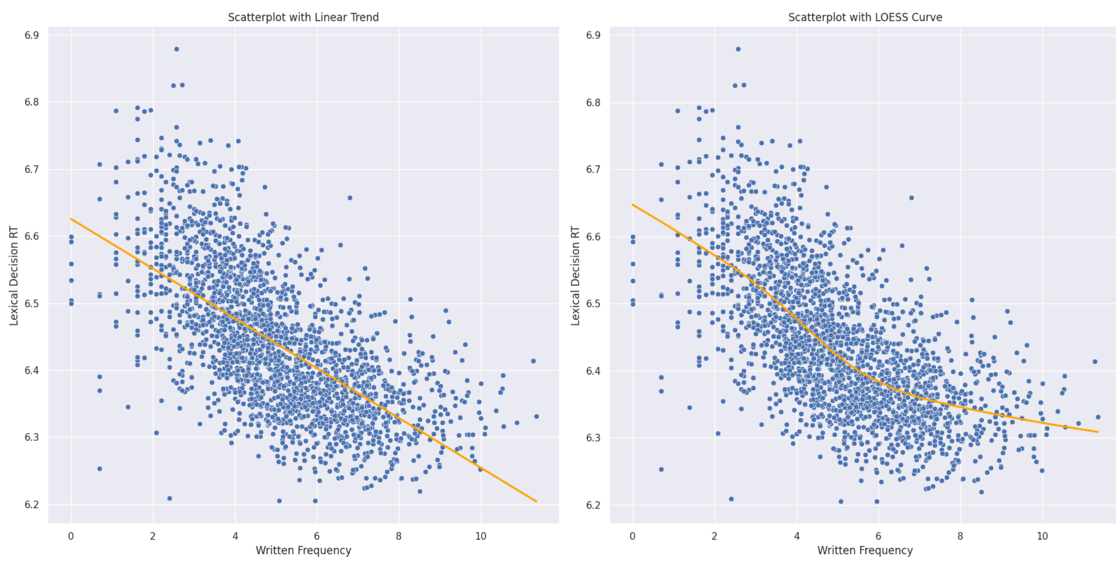
- In the right plot, put a scatterplot of written frequency (x-axis) and lexical decision RT (y-axis), with a superimposed LOESS of best fit.
- In both plots: adjust the size, transparency, and/or color of the lines and/or dots as appropriate.

You may find the Seaborne help pages useful, such as this one. Some possible functions to use:

- `plt` and `plt.subplots` from `matplotlib.pyplot`
- `regplot` from `seaborn`

[11]:
```
#Problem 3:

fig, axes = plt.subplots(1, 2, figsize=(18, 9))

# Plot 1: linear trend line
sns.scatterplot(ax=axes[0], x='WrittenFrequency', y='RTlexdec',␣
↪data=english_young)
sns.regplot(ax=axes[0], x='WrittenFrequency', y='RTlexdec', data=english_young,␣
↪scatter=False, ci=None, color='orange')

axes[0].set_title('Scatterplot with Linear Trend')
axes[0].set_xlabel('Written Frequency')
axes[0].set_ylabel('Lexical Decision RT')

# Plot 2: LOESS curve
```

```
sns.scatterplot(ax=axes[1], x='WrittenFrequency', y='RTlexdec',␣
↪data=english_young)
sns.regplot(ax=axes[1], x='WrittenFrequency', y='RTlexdec', data=english_young,␣
↪scatter=False, lowess=True, ci=None, color='orange')

axes[1].set_title('Scatterplot with LOESS Curve')
axes[1].set_xlabel('Written Frequency')
axes[1].set_ylabel('Lexical Decision RT')

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



2.7 Question 5 (2 points)

Based on these two plots, do you think that a linear model represents the relationship between written frequency and reaction time? Why/why not? If we fit a polynomial approximation of order k to the LOESS curve, what k do you think would be most appropriate? You can specify up to two possible k values (e.g. “k = 3” or “k = 1–2” is OK, “k = 3, 5 or 9” is not). Your answer should be verbal, with your guess at k purely based on visual inspection.

NB: A line is a polynomial.

The linear model does not appear to fully capture the relationship between written frequency and lexical decision RT, as the data does not closely follow the straight trend line as the written frequency gets very high. Observing the LOESS curve suggests a non-linear relationship, which could be better represented by a polynomial model. A polynomial of order 2 or 3, allowing for curvature and the inflection point ($\approx$ written frequency = 6), would likely be a more appropriate fit based on visual inspection.

2.8 Question 6 (2 points)

When modeling any relationship in data, it's important to think not just about what quantitative model (e.g. a line vs. a LOESS curve) fits best, but what relationships are possible given domain-specific knowledge.

Let's consider the linear fit from this perspective. Think about what a linear fit predicts for reaction time as written frequency is changed, and what people are doing in a lexical decision task. Is there any issue (or multiple issues) that tells us that the true relationship cannot be linear? Explain.

A linear fit implies that reaction times would consistently decrease with written frequency, which is not necessarily true. An average human's reaction time will likely plateau at some point due to biological constraints, where even further increases in written frequency will not be able to break it.

3 Problem 6 (2 points), Problem 7 (4 points)

We'll now check your intuition from above by examining more complex models of the relationship between frequency and lexical decision time, similarly to cases in the Regression CoLab notebook considered in class.

Fill in the following code for fitting polynomial regressions of degree k, choosing the best k, and visualizing the resulting relationship.

The one difference from the code considered in class is that we will consider two measures of goodness of fit:

1. R^2 on the test set
2. Bayesian Information Criterion (BIC) on the test set

Note that as defined here, higher BIC = better model.

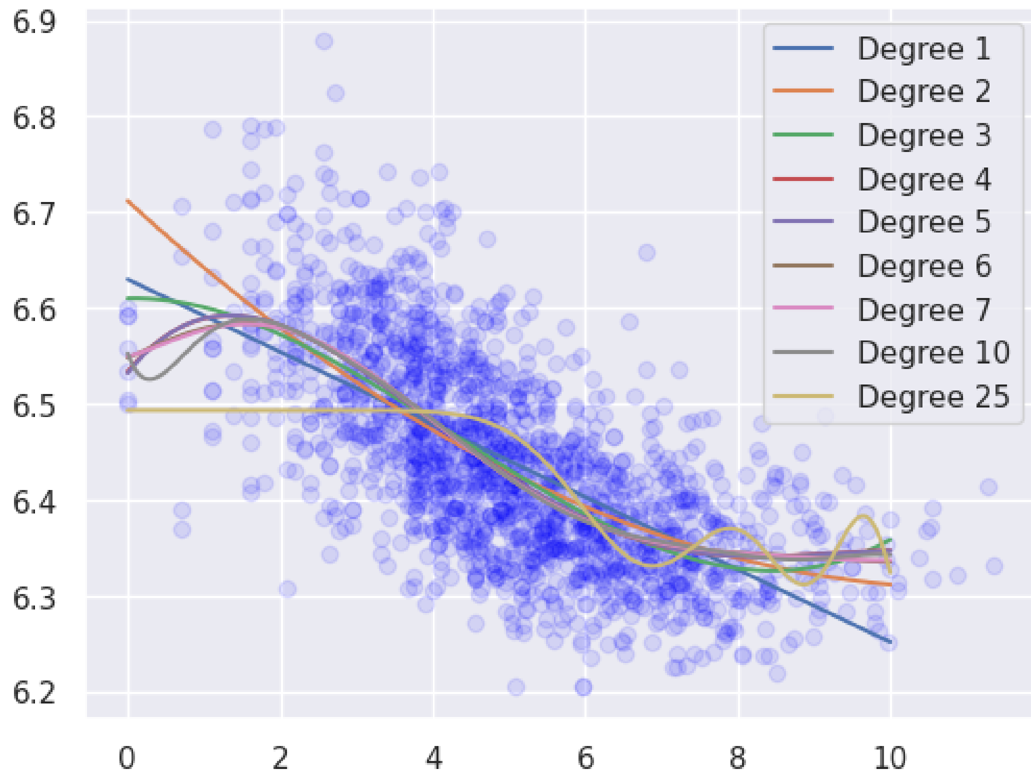
Hint: Do not implement your own function for train/test splitting, or for computing polynomial components.

[12]:
```
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import numpy as np

from sklearn.metrics import mean_squared_error

def bic(X, y, model):
  # number of observations
  n = X.shape[0]

  # number of parameters
  k = X.shape[1] + 1
```

```
  # calculate Residual Sum of Squares
  RSS = mean_squared_error(y, model.predict(X)) * n

  BIC = n * np.log(RSS / n) + k * np.log(n)

  return(BIC)



# Problem 6:  preprocessing

# Define the features and outcome

X = english_young['WrittenFrequency'].values.reshape(-1, 1)
y = english_young['RTlexdec']


# - Split the data into train and test subsets, with 20% of the data in test.
# This should define objects called X_train, X_test, y_train, and y_test.

X_train, X_test, y_train, y_test = train_test_split(X,y)

######
# Sets up a scatterplot of training data:
X_plot = np.linspace(0, 10,5000).reshape(-1, 1)
plt.scatter(X_train, y_train, color='blue', alpha=0.1)

######

## Problem 7: polynomial regression + visualization

print("Model class: " + "Linear Regression")
for degree in [1,2,3,4,5,6,7,10,25]:
  # - fit a polynomial regression model with this degree, on the training data
  # it should be named 'model'.

    model = make_pipeline(PolynomialFeatures(degree), LinearRegression())
    model.fit(X_train, y_train)

    print("\tDegree " + str(degree) + "\n\t\tTrain R^2: " + str(model.
 ↪score(X_train, y_train)))
    print("\t\tTest R^2: " + str(model.score(X_test, y_test)))
    print("\t\tBIC: " + str(bic(X_test, y_test, model)))

    y_plot = model.predict(X_plot)
    plt.plot(X_plot, y_plot, label=f'Degree {degree}')
```

```python
plt.legend()
plt.show()
```

```
Model class: Linear Regression
        Degree 1
                Train R^2: 0.4347136975190682
                Test R^2: 0.34974571406012245
                BIC: -2803.9263972212493
        Degree 2
                Train R^2: 0.45647176742453055
                Test R^2: 0.35661079620083147
                BIC: -2809.9867975277384
        Degree 3
                Train R^2: 0.47351474012166395
                Test R^2: 0.3726600381940146
                BIC: -2824.4109639762705
        Degree 4
                Train R^2: 0.48161632247504693
                Test R^2: 0.38264526756171413
                BIC: -2833.5725492395877
        Degree 5
                Train R^2: 0.4816204387675368
                Test R^2: 0.382645675583974
                BIC: -2833.57292662519
        Degree 6
                Train R^2: 0.48250545610330553
                Test R^2: 0.38478369903020526
                BIC: -2835.5538477782206
        Degree 7
                Train R^2: 0.4825184991977808
                Test R^2: 0.3850675448968035
                BIC: -2835.8173541093106
        Degree 10
                Train R^2: 0.48289319835194835
                Test R^2: 0.38688132719773294
                BIC: -2837.5040435693363
        Degree 25
                Train R^2: 0.3468320990153132
                Test R^2: 0.28372138324305285
                BIC: -2748.7074280791894
```



3.1 Question 7 (3 points)

Which degree polynomial provided the best fit to this dataset based on R^2? Based on BIC? Which answer makes more sense given your answers to Questions 5 and 6? Is the relationship between frequency and lexical decision time linear or nonlinear?

Based on the test R^2, the polynomial model of degree 7 ($R^2 \approx 0.4445$) provided the best fit to the dataset. This validates the previous answer, where we stated that at a high word frequency, we see a plateau in lexical decision reaction time. Therefore, the relationship bnetween frequency and lexical decision time is non-linear

3.2 Problem 8 (4 points)

Let's now fit a model using all predictors, including a polynomial effect of `WrittenFrequency`, of the degree you chose in Question 6.

For interpreting the model coefficients, it's useful to first standardize both y and the columns of X.

Prepare the data for this model:

Hint: Do not implement your own function for z-scoring each column of a DataFrame.

[13]:
```
# Problem 8
```

```python
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

## define X such that the columns are predictor variables in english_young,
## with columns added for polynomial features.
##
## for example, if you found in Question 6 that k = 4, then you'd add a
## columns here called WrittenFrequency2, which is the square of the␣
↪WrittenFrequency column,
## and similarly for WrittenFrequency3 and WrittenFrequency4

#t Your code here

k = 7

X = english_young.drop(columns=['RTlexdec', 'Word', 'AgeSubject'])

polynomial_features = PolynomialFeatures(degree=k, include_bias=False)
written_frequency_poly = polynomial_features.
↪fit_transform(english_young[['WrittenFrequency']])

poly_feature_names = [f'WrittenFrequency^{i}' for i in range(2, k+1)]

# Adding the polynomial features to the original predictors DataFrame
for i, name in enumerate(poly_feature_names, start=1):
    X[name] = written_frequency_poly[:, i]

y = english_young['RTlexdec']

## Now: define X_std and Y_std:
## - X_std is the X matrix above, but with each column z-scored
## - y_std is the same as y above, but z-scored

scaler = StandardScaler()

X_std = scaler.fit_transform(X)
X_std = pd.DataFrame(X_std, columns=X.columns)  # Convert back to DataFrame and␣
↪add column names

# Standardize the outcome variable
y_std = scaler.fit_transform(y.values.reshape(-1, 1)).flatten()  # Reshape y to␣
↪a 2D array for StandardScaler and flatten the result back to 1D

# - Split the data into train and test subsets, with 20% of the data in test.
# This should define objects called X_train, X_test, y_train, and y_test.
```

```python
X_train, X_test, y_train, y_test = train_test_split(X_std, y_std, test_size=0.
 ↪2, random_state=42)

# - Modeling
model = LinearRegression()

model.fit(X_train, y_train)

train_score = model.score(X_train, y_train)
test_score = model.score(X_test, y_test)

# Get the model coefficients
coefficients = model.coef_

# Get the standardized predicted values for the test set
y_pred_test = model.predict(X_test)

# You could also calculate the BIC for the model using the test data
# Using the previously defined bic function
bic_value = bic(X_test, y_test, model)

# Output the results
print(f'Train R^2: {train_score:.3f}')
print(f'Test R^2: {test_score:.3f}')
print(f'BIC: {bic_value:.3f}')
print('Model Coefficients:', coefficients)
```

```
Train R^2: 0.562
Test R^2: 0.449
BIC: -192.298
Model Coefficients: [-3.64068629e-01  1.33774508e-02  3.66264767e-01
5.59710310e-02
 -9.61839555e-02 -6.45704402e-02  8.26580337e-03 -2.43552283e-02
  1.84790124e+00 -1.86658470e+01  3.12557247e+01 -1.20686514e+01
 -9.06409825e+00  6.06406347e+00]
```

4 Problem 9 (3 points)

There are many predictors here, some of which probably don't actually have non-zero effects. We'll fit a Lasso regression, which should perform as well as linear regression, while allowing us to perform variable selection.

[14]:
```python
from sklearn.linear_model import Lasso

## Problem 9
```

```
# Fit a Lasso linear regression to X_std and y_std, with alpha parameter of 0.
↪02.
# (you can just assume this is a good alpha).  Call this model mod_lasso

# Initialize the Lasso regression model with an alpha parameter of 0.02
mod_lasso = Lasso(alpha=0.02)

# Fit the model on the standardized training data
mod_lasso.fit(X_train, y_train)

# We can check the coefficients of the model

# Print the R^2 of this model on the train and test set

train_score = model.score(X_train, y_train)
test_score = model.score(X_test, y_test)

print(f'Train R^2: {train_score:.3f}')
print(f'Test R^2: {test_score:.3f}')
```

```
Train R^2: 0.562
Test R^2: 0.449
```

4.1 Problem 10 (3 points)

Print out a pandas DataFrame summarizing the coefficient value for each predictor for this model, called `coefficients_with_features`. Column 1 should be predictor names and Column 2 coefficient values. The rows of the table should be sorted in order of coefficient magnitudes (= absolute values).

[15]:
```
## Problem 10

# extract model coeffs
lasso_coefficients = mod_lasso.coef_

# make the dataframe
lasso_coefficients_df = pd.DataFrame({'Feature': X_train.columns, 'Coefficient':
 ↪ lasso_coefficients})

# sort the DataFrame by coefficient magnitude
lasso_coefficients_df_sorted = lasso_coefficients_df.
↪reindex(lasso_coefficients_df.Coefficient.abs().sort_values(ascending=False).
↪index)

display(lasso_coefficients_df_sorted)
```

```
                        Feature  Coefficient
2              WrittenFrequency    -0.434607
```

```
0                    Familiarity    -0.371434
11           WrittenFrequency^5     0.108603
4                     FamilySize    -0.091300
12           WrittenFrequency^6     0.087170
5            InflectionalEntropy    -0.049765
3   WrittenSpokenFrequencyRatio     0.023153
7                          Voice    -0.003634
1                   WordCategory     0.000000
6                LengthInLetters     0.000000
8            WrittenFrequency^2    -0.000000
9            WrittenFrequency^3     0.000000
10           WrittenFrequency^4     0.000000
13           WrittenFrequency^7     0.000000
```

4.2 Question 8 (2 points)

According to the Lasso regression: * Which two predictors have the largest effects? * Which predictors are selected as having no effect?

Answer: The two predictors with the largest effects are Familiarity and WordCatgory. The foll-woingpredictors below had a coefficient of zero, implying having no effect: InflectionalEntropy, LengthInLEtters, Voice, WrittenFrequency^{2-4}

4.3 Question 9 (3 points)

You should find that two of the predictors that have large effects are very correlated (see the empirical plot above). Call these x_1 and x_2. What are x_1 and x_2? (Choose the most-correlated pair of predictors.)

Suppose that in reality, only x_1 (causally) affects `RTlexdec`, and x_2 just looks correlated with `RTlexdec` because it's highly correlated with x_1. Why hasn't Lasso selected x_2 as having no effect (and will not do so, even if we increase `alpha`)?

Answer: Lasso regression selects predictors based on their predictive power rather than causal relationships, particularly when variables are highly correlated. In the case of x_1 (written frequency) and x_2 (familiarity), both may be retained by Lasso because they significantly contribute to predicting RTlexdec, despite their correlations. Lasso may not distinguish between causally related and merely correlated predictors because it only measures predictive power and not the causal relationship.

4.4 Question 10 (3 points)

- Why is R^2 on the test set lower than on the training set?
- If the `alpha` parameter were increased, would we expect the R^2 on the test set to increase or decrease? Do we expect more or fewer predictors to be selected as having no effect? Explain.

Answer: R^2 on the test set is typically lower than on the training set because the model is optimized for the training data, leading to potential overfitting and less generalization to unseen data. Increasing *alpha* in Lasso regression generally leads to a decrease in R^2 on the test set, as stronger regularization can oversimplify the model, potentially underfitting the data. A higher alpha also results in more predictors being selected as having no effect, increasing sparsity in the model by penalizing the inclusion of less important variables.

5 Part 2: Classification with Pokémon data

This part uses Pokémon name data to examine sound symbolism: to what extent are properties of a Pokémon predictable from its name? We will be considering *evolution*, a fundamental division between Pokémon characters. For our purposes, Pokémon can be either *evolved* or *non*-evolved. (The real story is more complicated, as many of you know, but this is a reasonable first approximation.)

An interesting aspect of Pokémon for linguistic research is that complete Pokémon name sets exist in different languages, giving us multiple datasets to examine sound symbolism and to what extent it looks similar across languages.

In class we examined Pokémon evolution status as a classification problem for English names . In this homework, we'll do the same for Mandarin Chinese names (henceforth "Mandarin").

This data comes from a recent paper:

Kilpatrick, A., Ćwiek, A., and Kawahara, S. (2023). Random forests, sound symbolism and Pokémon evolution. PLoS ONE 18(1): e0279350. https://doi.org/10.1371/journal.pone.0279350

This paper considers Korean, Japanese, and Mandarin datasets, all available in this OSF project.

The datafile we are using, `chinese_pokemon.csv`, is derived from the data on this site.

Copy the data to your Drive folder from here.

[16]:
```
# throws an error if your Drive folder doesn't contain chinese_pokemon.csv
from google.colab import drive
drive.mount('/content/drive/')
!ls "/content/drive/My Drive/chinese_pokemon.csv"
```

```
Drive already mounted at /content/drive/; to attempt to forcibly remount, call
drive.mount("/content/drive/", force_remount=True).
'/content/drive/My Drive/chinese_pokemon.csv'
```

First, load the data and take a look:

[17]:
```
chinese = pd.read_csv("/content/drive/My Drive/chinese_pokemon.csv")
display(chinese)
```

```
              name  length  evolved  flat_tone  rising_tone  \
0      miàowāZǒNzǐ      11        0          1            0
1        miàowācǎo       9        1          1            0
2        miàowāhuā       9        1          2            0
3       xiǎohuǒlóN      10        0          0            1
4        huǒkǒNlóN       9        1          0            1
..             …       …        …          …            …
893   léijíHàilèqí      12        0          0            3
894   léijíduólāgē      12        0          2            3
895       xuěbàomǎ       8        0          0            0
```

```
896      líNyōumǎ       8        0          1            1
897    lěiguànwáN      10        0          0            1

     falling_rising_tone  falling_tone  neutral_tone  a  e  …  r  x  h  l  \
0                      2             1             2  2  0  …  0  0  0  0
1                      1             1             3  3  0  …  0  0  0  0
2                      0             1             3  3  0  …  0  0  1  0
3                      2             0             3  1  0  …  0  1  1  1
4                      2             0             1  0  0  …  0  0  1  1
..                   …             …             … .. ..  … .. .. .. ..
893                    0             2             2  1  2  …  0  0  0  2
894                    0             0             2  1  2  …  0  0  0  2
895                    2             1             2  2  1  …  0  1  0  0
896                    1             0             1  1  0  …  0  0  0  1
897                    1             1             2  2  1  …  0  0  0  1

     w  q  y  H  hyphen  colon
0    1  0  0  0       0      0
1    1  0  0  0       0      0
2    1  0  0  0       0      0
3    0  0  0  0       0      0
4    0  0  0  0       0      0
..  .. .. .. ..     …    …
893  0  1  0  1       0      0
894  0  0  0  0       0      0
895  0  0  0  0       0      0
896  0  0  1  0       0      0
897  1  0  0  0       0      0

[898 rows x 41 columns]
```

Column meanings:

- `name`: Pokémon's name, in a custom transcription system*
- `length`: length of name, in phones
- `evolved`: evolved Pokémon? 0/1 = False/True
- `flat_tone`, `rising_tone`, etc.: number of syllables in the name carrying this tone
- `a`, `i`, `e`, etc: number of times this phone appears in the name

The transcription system used is close to Pinyin, but modified so that every phoneme is represented by a single ASCII character—similar to the X-SAMPA system for English used in our `h95.csv` vowels dataset. (If you are curious / familiar with Mandarin, the system is described on p. 5 of this document.) Some things you may need to know for this homework are:

- Every syllable in Mandarin bears one of 5 tones: flat, rising, falling-rising, falling, or neutral.
- `U` stands for a front rounded vowel (written "ü" in Pinyin)
- `N` stands for the velar nasal, which can only occur at the end of syllables in Mandarin (written "ng" in Pinyin).
- `j` stands for the affricate /t / (written "j" in Pinyin).
- `y` stands for the glide /j/ (written "y" in Pinyin).

5.1 Problem 11 (2 points)

Prepare the data:

- Make the predictor matrix: a numpy DataFrame `X` which consists of all columns except `name` and `evolved`.
- Make the outcome vector `y`
- Split the data into train and test subsets, with 20% of the data in test. This should define objects called `X_train`, `X_test`, `y_train`, and `y_test`.

[18]:
```
# Problem 11
from sklearn.model_selection import train_test_split

X = chinese.drop(columns=['name', 'evolved'])
y = chinese['evolved']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=RANDOM_STATE)
```

We will fit two classification models to this dataset, with the goal of determining which predictors (properties of a Pokémon's name) affect `evolution`.

Some background on sound symbolism will be useful. We might hypothesize that "evolved" status would be correlated with some types of sounds which have been found to evoke large size/heaviness/hardness across languages:

- Low vowels, such as `a`: positive correlation
- High vowels, such as `i`: negative correlation
- Back vowels, such as `u`: negative correlation
- Nasal consonants, especially in syllable codas: positive correlation
- Bilabial consonants: negative correlation

One theory underlying such associations is Ohala's *frequency code hypothesis*, which posits that sounds that tend to have higher f0 (pitch) are more associated with greater size/weight/male gender.

For Pokémon names, it is well known (by players) that:

- *longer names* are positively correlated with "evolved" status (as well as higher power).

This pattern seems to be Pokémon-specific sound symbolism.

Interestingly, not much is known about sound symbolism involving tones across languages, including in Mandarin Chinese (the world's most-spoken tone language).

5.2 Problem 12 (3 points)

We will first fit and evaluate a logistic regression model to predict `evolved`.

[22]:
```
# Problem 12

from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

```python
# Fit a logistic regression model, called lr_model, to X_train and y_train.
# Make sure that the model does not use any regularization -- the
# sklearn default includes L2 regularization.

lr_model = LogisticRegression(penalty='none', solver='lbfgs', max_iter=10000)
lr_model.fit(X_train, y_train)

# Calculate the accuracy on the training set and on the test set.
# save these as train_acc and test_acc

train_predictions = lr_model.predict(X_train)
train_acc = accuracy_score(y_train, train_predictions)

test_predictions = lr_model.predict(X_test)
test_acc = accuracy_score(y_test, test_predictions)

# print these accuracies:
print([train_acc, test_acc])
```

```
[0.6183844011142061, 0.6111111111111112]
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:1173:
FutureWarning: `penalty='none'`has been deprecated in 1.2 and will be removed in
1.4. To keep the past behaviour, set `penalty=None`.
  warnings.warn(
```

5.3 Problem 13 (3 points)

To examine which predictors are important, print a table of coefficients where: * Each row corresponds to one predictor * Column 1: predictor names * Column 2: coeficient values * Rows sorted by decreasing coefficient *absolute value*.

Fill in the missing parts of code below.

[23]:
```python
## Problem 13

# make 'feature_names' the names of columns of X_train
# make 'coefficients' a numpy array consisting of the values of the
# coefficients of lr_model

feature_names = X_train.columns
coefficients = lr_model.coef_[0]

coef_table = pd.DataFrame({
    'Feature': feature_names,
    'Coefficient': coefficients
})
```

```
# sort the Dataframe by absolute value of coefficients, then print it.

coef_table = coef_table.reindex(coef_table.Coefficient.abs().
 ↪sort_values(ascending=False).index)

display(coef_table)
```

	Feature	Coefficient
37	hyphen	7.823728
38	colon	-7.058075
25	j	0.895046
14	N	0.803622
11	U	-0.772097
31	h	0.755377
23	z	0.674746
26	f	0.662834
33	w	0.648645
24	Z	0.611854
22	C	0.573916
28	S	0.551838
18	b	0.542945
17	k	0.533661
13	n	0.531180
19	d	0.519793
36	H	0.465347
20	g	0.424330
21	c	0.423946
32	l	0.419720
12	m	0.355755
35	y	0.353594
10	u	0.288419
4	falling_tone	0.239724
6	a	0.221274
30	x	-0.216083
7	e	0.206642
0	length	-0.199483
27	s	0.190105
16	t	0.175895
9	o	0.151392
8	i	0.133327
34	q	0.120131
1	flat_tone	0.101127
5	neutral_tone	-0.087385
15	p	0.085204
3	falling_rising_tone	-0.025756
29	r	0.022581
2	rising_tone	0.001248

5.4 Question 11 (2 points)

What are the four most important features, going by coefficient values? Briefly describe what they mean (e.g. `a` would be "number of times 'a' appears in the name").

Q10: put your answer here, with each feature one line of an itemized list. (4 sentences)

Based on the coefficient values, the four most important features are:

- hyphen: The number of hyphens (-) appearing in the name.
- colon: The number of colons (:) appearing in the name.
- j: The number of times the letter 'j' appears in the name.
- N: Represents the number of times the letter 'N' (capitalized) appears in the name.

6 Problem 14 (3 points)

Our second model will be a random forest. Fit a random forest called `rf` to the training data, with the following options:

- Minimum number of samples per split: 5
- Maximum tree depth: 5
- Use OOB score instead of accuracy
- Use 1000 trees

(These options make this particular random forest perform better, and you can just take them as given.)

Then print its accuracy on the train and test set.

[24]:
```python
## Problem 14

from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(n_estimators=1000,
                            min_samples_split=5,
                            max_depth=5,
                            oob_score=True,
                            random_state=42)

rf.fit(X_train, y_train)

# Evaluate the model
# OOB
print(f"OOB Score: {rf.oob_score_}")

# Accurrafcy
train_predictions = rf.predict(X_train)
test_predictions = rf.predict(X_test)
```

```python
train_accuracy = accuracy_score(y_train, train_predictions)
test_accuracy = accuracy_score(y_test, test_predictions)

print(f"Train Accuracy: {train_accuracy}")
print(f"Test Accuracy: {test_accuracy}")
```

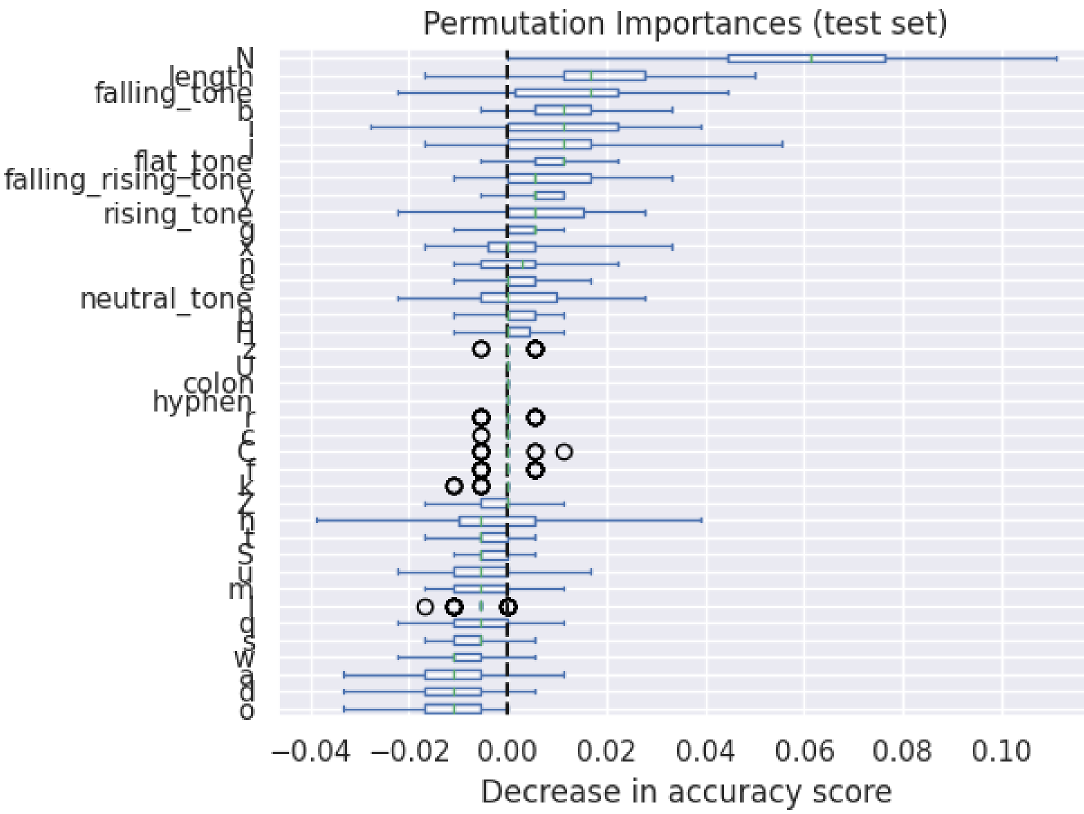
```
OOB Score: 0.5348189415041783
Train Accuracy: 0.7799442896935933
Test Accuracy: 0.5944444444444444
```

To compute feature importances for this random forest, we'll work from this sklearn vignette.

We will compute *permutation importance* on a held-out test set, as in the example shown after the paragraph beginning with "As an alternative, the permutation importances of rf are computed on a held out test set…" Read as much of the vignette as necessary to understand what is being done here, and what the boxplots in the following plot mean. (Why does the figure show a range of values for each feature, rather than just a single importance number?)

We adapt the code there to calculate permutation importance and show a plot of horizontal boxplots, like the one shown there:

[25]:
```python
from sklearn.inspection import permutation_importance

# This code will work after you've defined rf, but will take a while to run

## calculate permutation importance
result = permutation_importance(
    rf, X_test, y_test, n_repeats=50, random_state=42, n_jobs=-1
)

## arrange as a dataframe, sorted by importance
sorted_importances_idx = result.importances_mean.argsort()
importances = pd.DataFrame(
    result.importances[sorted_importances_idx].T,
    columns=X.columns[sorted_importances_idx],
)

# plot importances on the test set
ax = importances.plot.box(vert=False, whis=10)
ax.set_title("Permutation Importances (test set)")
ax.axvline(x=0, color="k", linestyle="--")
ax.set_xlabel("Decrease in accuracy score")
ax.figure.tight_layout()
```



6.1 Question 12 (2 points)

What are the four most important features, going by this plot? How much do these features overlap with those from Question 9?

Based on the coefficient values, the four most important features are: N, length, failling_tone, b. Only the feature `N` overlaps with the logistic regression model.

7 Problem 15 (4 points, up to 4 points extra credit)

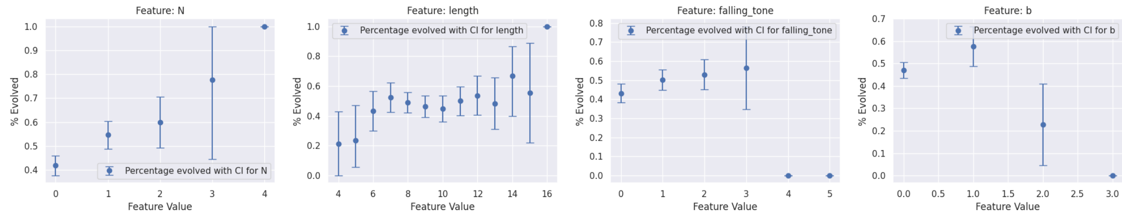
To get a sense of how each of these features affects `evolved`: for each feature, make four empirical plots: one for each feature, with the feature on the x-axis and % evolved on the y-axis. These plots should be in a 1x4 grid.

Each plot can just show one point per value of the feature, corresponding to the % of the data with this feature value (e.g. `a` = 2) for which `evolved` is 1.

Your plots should be **legible**, following the guidelines in Problem 3, though it's not required to show the empirical data in the plots.

Extra credit: calculate the error for each % evolved, and showing these on the plots (using 95% confidence intervals). Add information to the plot showing the empirical data: the number of points with `evolved` = 1 vs. 0 for each feature value. Just using a default scatterplot isn't informative (why?).

[26]:
```python
## Problem 15
from scipy.stats import binom

def calculate_percentage_and_ci(df, feature):
    # Group by feature value and calculate % evolved and confidence intervals
    summaries = df.groupby(feature).agg(
        evolved_count=('evolved', sum),
        total_count=('evolved', 'size')
    ).assign(
        percentage_evolved=lambda x: x.evolved_count / x.total_count
    )

    # confidence interval
    summaries['lower_ci'], summaries['upper_ci'] = zip(*summaries.apply(
        lambda row: binom.interval(confidence=0.95, n=row['total_count'],␣
↪p=row['percentage_evolved']) / row['total_count'],
        axis=1
    ))

    return summaries


# top four important features from the previous question
features = ['N', 'length', 'falling_tone', 'b']

summaries_list = [calculate_percentage_and_ci(chinese, f) for f in features]

fig, axes = plt.subplots(1, 4, figsize=(20, 4))

for ax, summary, feature in zip(axes, summaries_list, features):
    ax.errorbar(summary.index, summary['percentage_evolved'],
                yerr=[summary['percentage_evolved'] - summary['lower_ci'],
                      summary['upper_ci'] - summary['percentage_evolved']],
                fmt='o', capsize=5, label=f'Percentage evolved with CI for␣
↪{feature}')
    ax.set_title(f'Feature: {feature}')
    ax.set_xlabel('Feature Value')
    ax.set_ylabel('% Evolved')
    ax.legend()

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



7.1 Question 13 (4 points)

Using your plots from Problem 14 and the results of Question 12, discuss your findings from the random forest with respect to the sound symbolism background above. Be sure to consider at least one feature you do *not* find to be informative.

Answer: The plots suggests that the `length` and `N` features have a potential positive correlation with whether the pokemon is evolved, showing a general but variable upward trend as the feature value increases, though it is worthy to note that the variance is a lot higher with `N`.

In contrast, `falling_tone` and `b` do not exhibit a consistent and helpful trend, with the latter particularly showing a decrease in `evolved` percentage at the highest feature value.

Out of all 4 features, none of them correspond to a stable, high probability prediction of whether the pokemon is evolved from the single feature alone.

7.2 Extra Credit Problem/Question (up to 5 points)

You should find that the most-informative features are quite different for the logistic regression and random forest models. For the top two features listed as informative by the logistic regression model but not the RF model:

- Figure out why the LR but not the RF model has chosen them as informative.
- Explain why the RF model *doesn't* choose them as informative.
- Explain why the RF's behavior is preferable.

A full answer will require writing both code and prose.

[27]:
```python
lr_coef = lr_model.coef_[0]
rf_importances = rf.feature_importances_ # using the built in feature␣
  ↪importances fiedl

sorted_indices_lr = np.argsort(np.abs(lr_coef))[::-1]
sorted_indices_rf = np.argsort(rf_importances)[::-1]



sorted_features_lr = [X_train.columns[i] for i in sorted_indices_lr]
sorted_features_rf = [X_train.columns[i] for i in sorted_indices_rf]

print(f"Top two features for LR: {sorted_features_lr[:2]}")
print(f"Top two features for RF: {sorted_features_rf[:2]}")
```

```
Top two features for LR: ['hyphen', 'colon']
Top two features for RF: ['length', 'N']
```

Answer: Features may be informative in LR due to a strong linear association with the outcome, whereas RF may capture non-linear relationships missed by LR.

Additionally, if the relationship is not consistently helpful across different splits in the data, it may not be as important in RF.

The RF model is often preferable because it does not assume linearity and can handle complex interactions between features, making it potentially more robust to unseen data.

8 To Submit

To submit: * Name this notebook `YOUR_STUDENT_ID_Assignment_4.ipynb` and download it. * Convert this `.ipynb` file to a `.pdf` (e.g., using the following instructions).
* Upload the PDF to the Gradescope assignment "Assignment 4".
* Submit the `.ipynb` file on myCourses under Assignment 4.

(Note: `Print > Save as PDF` **will not work** because it will not display your figures correctly.)

You can convert the notebook to a PDF using the following instructions.

9 Converting this notebook to a PDF

1. Make sure you have renamed the notebook, e.g. `000000000_Assignment_4.ipynb` where `000000000` is your student ID.
2. Make sure to save the notebook (`ctrl/cmd + s`).

2. Make sure Google Drive is mounted (it likely already is from the first question).

[28]:
```
from google.colab import drive
drive.mount('/content/drive/')
!ls "/content/drive/MyDrive/Colab Notebooks/"
```

```
Drive already mounted at /content/drive/; to attempt to forcibly remount, call
drive.mount("/content/drive/", force_remount=True).
 261072449_Assignment_1.ipynb      COMP579_A2_2024_template.ipynb
 261072449_Assignment_2.ipynb      NL2DS_Winter_2024_Assignment_4.pdf
 261072449_Assignment_4.ipynb      tweets
 261072449_Assignment_4.pdf        Untitled0.ipynb
'Assignment 1 Winter 2024.ipynb'
```

3. Install packages for converting .ipynb to .pdf

[ ]:
```
!apt-get -q install texlive-xetex texlive-fonts-recommended
 ↪texlive-plain-generic
```

```
Reading package lists…
Building dependency tree…
Reading state information…
```

```
The following additional packages will be installed:
  dvisvgm fonts-droid-fallback fonts-lato fonts-lmodern fonts-noto-mono fonts-
texgyre
  fonts-urw-base35 libapache-pom-java libcommons-logging-java libcommons-parent-
java
  libfontbox-java libfontenc1 libgs9 libgs9-common libidn12 libijs-0.35
libjbig2dec0 libkpathsea6
  libpdfbox-java libptexenc1 libruby3.0 libsynctex2 libteckit0 libtexlua53
libtexluajit2 libwoff1
  libzzip-0-13 lmodern poppler-data preview-latex-style rake ruby ruby-net-
telnet ruby-rubygems
  ruby-webrick ruby-xmlrpc ruby3.0 rubygems-integration t1utils teckit tex-
common tex-gyre
  texlive-base texlive-binaries texlive-latex-base texlive-latex-extra texlive-
latex-recommended
  texlive-pictures tipa xfonts-encodings xfonts-utils
Suggested packages:
  fonts-noto fonts-freefont-otf | fonts-freefont-ttf libavalon-framework-java
  libcommons-logging-java-doc libexcalibur-logkit-java liblog4j1.2-java poppler-
utils ghostscript
  fonts-japanese-mincho | fonts-ipafont-mincho fonts-japanese-gothic | fonts-
ipafont-gothic
  fonts-arphic-ukai fonts-arphic-uming fonts-nanum ri ruby-dev bundler debhelper
gv
  | postscript-viewer perl-tk xpdf | pdf-viewer xzdec texlive-fonts-recommended-
doc
  texlive-latex-base-doc python3-pygments icc-profiles libfile-which-perl
  libspreadsheet-parseexcel-perl texlive-latex-extra-doc texlive-latex-
recommended-doc
  texlive-luatex texlive-pstricks dot2tex prerex texlive-pictures-doc vprerex
default-jre-headless
  tipa-doc
The following NEW packages will be installed:
  dvisvgm fonts-droid-fallback fonts-lato fonts-lmodern fonts-noto-mono fonts-
texgyre
  fonts-urw-base35 libapache-pom-java libcommons-logging-java libcommons-parent-
java
  libfontbox-java libfontenc1 libgs9 libgs9-common libidn12 libijs-0.35
libjbig2dec0 libkpathsea6
  libpdfbox-java libptexenc1 libruby3.0 libsynctex2 libteckit0 libtexlua53
libtexluajit2 libwoff1
  libzzip-0-13 lmodern poppler-data preview-latex-style rake ruby ruby-net-
telnet ruby-rubygems
  ruby-webrick ruby-xmlrpc ruby3.0 rubygems-integration t1utils teckit tex-
common tex-gyre
  texlive-base texlive-binaries texlive-fonts-recommended texlive-latex-base
texlive-latex-extra
  texlive-latex-recommended texlive-pictures texlive-plain-generic texlive-xetex
```

```
tipa
  xfonts-encodings xfonts-utils
0 upgraded, 54 newly installed, 0 to remove and 38 not upgraded.
Need to get 182 MB of archives.
After this operation, 571 MB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy/main amd64 fonts-droid-fallback all
1:6.0.1r16-1.1build1 [1,805 kB]
Get:2 http://archive.ubuntu.com/ubuntu jammy/main amd64 fonts-lato all 2.0-2.1
[2,696 kB]
Get:3 http://archive.ubuntu.com/ubuntu jammy/main amd64 poppler-data all
0.4.11-1 [2,171 kB]
Get:4 http://archive.ubuntu.com/ubuntu jammy/universe amd64 tex-common all 6.17
[33.7 kB]
Get:5 http://archive.ubuntu.com/ubuntu jammy/main amd64 fonts-urw-base35 all
20200910-1 [6,367 kB]
Get:6 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libgs9-common
all 9.55.0~dfsg1-0ubuntu5.6 [751 kB]
Get:7 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libidn12 amd64
1.38-4ubuntu1 [60.0 kB]
Get:8 http://archive.ubuntu.com/ubuntu jammy/main amd64 libijs-0.35 amd64
0.35-15build2 [16.5 kB]
Get:9 http://archive.ubuntu.com/ubuntu jammy/main amd64 libjbig2dec0 amd64
0.19-3build2 [64.7 kB]
Get:10 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libgs9 amd64
9.55.0~dfsg1-0ubuntu5.6 [5,031 kB]
Get:11 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libkpathsea6
amd64 2021.20210626.59705-1ubuntu0.1 [60.3 kB]
Get:12 http://archive.ubuntu.com/ubuntu jammy/main amd64 libwoff1 amd64
1.0.2-1build4 [45.2 kB]
Get:13 http://archive.ubuntu.com/ubuntu jammy/universe amd64 dvisvgm amd64
2.13.1-1 [1,221 kB]
Get:14 http://archive.ubuntu.com/ubuntu jammy/universe amd64 fonts-lmodern all
2.004.5-6.1 [4,532 kB]
Get:15 http://archive.ubuntu.com/ubuntu jammy/main amd64 fonts-noto-mono all
20201225-1build1 [397 kB]
Get:16 http://archive.ubuntu.com/ubuntu jammy/universe amd64 fonts-texgyre all
20180621-3.1 [10.2 MB]
Get:17 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libapache-pom-java
all 18-1 [4,720 B]
Get:18 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libcommons-parent-
java all 43-1 [10.8 kB]
Get:19 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libcommons-logging-
java all 1.2-2 [60.3 kB]
Get:20 http://archive.ubuntu.com/ubuntu jammy/main amd64 libfontenc1 amd64
1:1.1.4-1build3 [14.7 kB]
Get:21 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libptexenc1
amd64 2021.20210626.59705-1ubuntu0.1 [39.1 kB]
Get:22 http://archive.ubuntu.com/ubuntu jammy/main amd64 rubygems-integration
```

```
all 1.18 [5,336 B]
Get:23 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 ruby3.0 amd64
3.0.2-7ubuntu2.4 [50.1 kB]
Get:24 http://archive.ubuntu.com/ubuntu jammy/main amd64 ruby-rubygems all
3.3.5-2 [228 kB]
Get:25 http://archive.ubuntu.com/ubuntu jammy/main amd64 ruby amd64 1:3.0~exp1
[5,100 B]
Get:26 http://archive.ubuntu.com/ubuntu jammy/main amd64 rake all 13.0.6-2 [61.7
kB]
Get:27 http://archive.ubuntu.com/ubuntu jammy/main amd64 ruby-net-telnet all
0.1.1-2 [12.6 kB]
Get:28 http://archive.ubuntu.com/ubuntu jammy/universe amd64 ruby-webrick all
1.7.0-3 [51.8 kB]
Get:29 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 ruby-xmlrpc all
0.3.2-1ubuntu0.1 [24.9 kB]
Get:30 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libruby3.0
amd64 3.0.2-7ubuntu2.4 [5,113 kB]
Get:31 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libsynctex2
amd64 2021.20210626.59705-1ubuntu0.1 [55.5 kB]
Get:32 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libteckit0 amd64
2.5.11+ds1-1 [421 kB]
Get:33 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libtexlua53
amd64 2021.20210626.59705-1ubuntu0.1 [120 kB]
Get:34 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libtexluajit2
amd64 2021.20210626.59705-1ubuntu0.1 [267 kB]
Get:35 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libzzip-0-13 amd64
0.13.72+dfsg.1-1.1 [27.0 kB]
Get:36 http://archive.ubuntu.com/ubuntu jammy/main amd64 xfonts-encodings all
1:1.0.5-0ubuntu2 [578 kB]
Get:37 http://archive.ubuntu.com/ubuntu jammy/main amd64 xfonts-utils amd64
1:7.7+6build2 [94.6 kB]
Get:38 http://archive.ubuntu.com/ubuntu jammy/universe amd64 lmodern all
2.004.5-6.1 [9,471 kB]
Get:39 http://archive.ubuntu.com/ubuntu jammy/universe amd64 preview-latex-style
all 12.2-1ubuntu1 [185 kB]
Get:40 http://archive.ubuntu.com/ubuntu jammy/main amd64 t1utils amd64
1.41-4build2 [61.3 kB]
Get:41 http://archive.ubuntu.com/ubuntu jammy/universe amd64 teckit amd64
2.5.11+ds1-1 [699 kB]
Get:42 http://archive.ubuntu.com/ubuntu jammy/universe amd64 tex-gyre all
20180621-3.1 [6,209 kB]
Get:43 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 texlive-
binaries amd64 2021.20210626.59705-1ubuntu0.1 [9,848 kB]
Get:44 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-base all
2021.20220204-1 [21.0 MB]
Get:45 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-fonts-
recommended all 2021.20220204-1 [4,972 kB]
Get:46 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-latex-base
```

```
all 2021.20220204-1 [1,128 kB]
Get:47 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libfontbox-java all
1:1.8.16-2 [207 kB]
Get:48 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libpdfbox-java all
1:1.8.16-2 [5,199 kB]
Get:49 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-latex-
recommended all 2021.20220204-1 [14.4 MB]
Get:50 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-pictures
all 2021.20220204-1 [8,720 kB]
Get:51 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-latex-extra
all 2021.20220204-1 [13.9 MB]
Get:52 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-plain-
generic all 2021.20220204-1 [27.5 MB]
Get:53 http://archive.ubuntu.com/ubuntu jammy/universe amd64 tipa all 2:1.3-21
[2,967 kB]
Get:54 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-xetex all
2021.20220204-1 [12.4 MB]
Fetched 182 MB in 17s (10.6 MB/s)
Extracting templates from packages: 100%
Preconfiguring packages …
Selecting previously unselected package fonts-droid-fallback.
(Reading database … 121752 files and directories currently installed.)
Preparing to unpack …/00-fonts-droid-fallback_1%3a6.0.1r16-1.1build1_all.deb
…
Unpacking fonts-droid-fallback (1:6.0.1r16-1.1build1) …
Selecting previously unselected package fonts-lato.
Preparing to unpack …/01-fonts-lato_2.0-2.1_all.deb …
Unpacking fonts-lato (2.0-2.1) …
Selecting previously unselected package poppler-data.
Preparing to unpack …/02-poppler-data_0.4.11-1_all.deb …
Unpacking poppler-data (0.4.11-1) …
Selecting previously unselected package tex-common.
Preparing to unpack …/03-tex-common_6.17_all.deb …
Unpacking tex-common (6.17) …
Selecting previously unselected package fonts-urw-base35.
Preparing to unpack …/04-fonts-urw-base35_20200910-1_all.deb …
Unpacking fonts-urw-base35 (20200910-1) …
Selecting previously unselected package libgs9-common.
Preparing to unpack …/05-libgs9-common_9.55.0~dfsg1-0ubuntu5.6_all.deb …
Unpacking libgs9-common (9.55.0~dfsg1-0ubuntu5.6) …
Selecting previously unselected package libidn12:amd64.
Preparing to unpack …/06-libidn12_1.38-4ubuntu1_amd64.deb …
Unpacking libidn12:amd64 (1.38-4ubuntu1) …
Selecting previously unselected package libijs-0.35:amd64.
Preparing to unpack …/07-libijs-0.35_0.35-15build2_amd64.deb …
Unpacking libijs-0.35:amd64 (0.35-15build2) …
Selecting previously unselected package libjbig2dec0:amd64.
Preparing to unpack …/08-libjbig2dec0_0.19-3build2_amd64.deb …
```

```
Unpacking libjbig2dec0:amd64 (0.19-3build2) …
Selecting previously unselected package libgs9:amd64.
Preparing to unpack …/09-libgs9_9.55.0~dfsg1-0ubuntu5.6_amd64.deb …
Unpacking libgs9:amd64 (9.55.0~dfsg1-0ubuntu5.6) …
Selecting previously unselected package libkpathsea6:amd64.
Preparing to unpack …/10-libkpathsea6_2021.20210626.59705-1ubuntu0.1_amd64.deb
…
Unpacking libkpathsea6:amd64 (2021.20210626.59705-1ubuntu0.1) …
Selecting previously unselected package libwoff1:amd64.
Preparing to unpack …/11-libwoff1_1.0.2-1build4_amd64.deb …
Unpacking libwoff1:amd64 (1.0.2-1build4) …
Selecting previously unselected package dvisvgm.
Preparing to unpack …/12-dvisvgm_2.13.1-1_amd64.deb …
Unpacking dvisvgm (2.13.1-1) …
Selecting previously unselected package fonts-lmodern.
Preparing to unpack …/13-fonts-lmodern_2.004.5-6.1_all.deb …
Unpacking fonts-lmodern (2.004.5-6.1) …
Selecting previously unselected package fonts-noto-mono.
Preparing to unpack …/14-fonts-noto-mono_20201225-1build1_all.deb …
Unpacking fonts-noto-mono (20201225-1build1) …
Selecting previously unselected package fonts-texgyre.
Preparing to unpack …/15-fonts-texgyre_20180621-3.1_all.deb …
Unpacking fonts-texgyre (20180621-3.1) …
Selecting previously unselected package libapache-pom-java.
Preparing to unpack …/16-libapache-pom-java_18-1_all.deb …
Unpacking libapache-pom-java (18-1) …
Selecting previously unselected package libcommons-parent-java.
Preparing to unpack …/17-libcommons-parent-java_43-1_all.deb …
Unpacking libcommons-parent-java (43-1) …
Selecting previously unselected package libcommons-logging-java.
Preparing to unpack …/18-libcommons-logging-java_1.2-2_all.deb …
Unpacking libcommons-logging-java (1.2-2) …
Selecting previously unselected package libfontenc1:amd64.
Preparing to unpack …/19-libfontenc1_1%3a1.1.4-1build3_amd64.deb …
Unpacking libfontenc1:amd64 (1:1.1.4-1build3) …
Selecting previously unselected package libptexenc1:amd64.
Preparing to unpack …/20-libptexenc1_2021.20210626.59705-1ubuntu0.1_amd64.deb
…
Unpacking libptexenc1:amd64 (2021.20210626.59705-1ubuntu0.1) …
Selecting previously unselected package rubygems-integration.
Preparing to unpack …/21-rubygems-integration_1.18_all.deb …
Unpacking rubygems-integration (1.18) …
Selecting previously unselected package ruby3.0.
Preparing to unpack …/22-ruby3.0_3.0.2-7ubuntu2.4_amd64.deb …
Unpacking ruby3.0 (3.0.2-7ubuntu2.4) …
Selecting previously unselected package ruby-rubygems.
Preparing to unpack …/23-ruby-rubygems_3.3.5-2_all.deb …
Unpacking ruby-rubygems (3.3.5-2) …
```

```
Selecting previously unselected package ruby.
Preparing to unpack …/24-ruby_1%3a3.0~exp1_amd64.deb …
Unpacking ruby (1:3.0~exp1) …
Selecting previously unselected package rake.
Preparing to unpack …/25-rake_13.0.6-2_all.deb …
Unpacking rake (13.0.6-2) …
Selecting previously unselected package ruby-net-telnet.
Preparing to unpack …/26-ruby-net-telnet_0.1.1-2_all.deb …
Unpacking ruby-net-telnet (0.1.1-2) …
Selecting previously unselected package ruby-webrick.
Preparing to unpack …/27-ruby-webrick_1.7.0-3_all.deb …
Unpacking ruby-webrick (1.7.0-3) …
Selecting previously unselected package ruby-xmlrpc.
Preparing to unpack …/28-ruby-xmlrpc_0.3.2-1ubuntu0.1_all.deb …
Unpacking ruby-xmlrpc (0.3.2-1ubuntu0.1) …
Selecting previously unselected package libruby3.0:amd64.
Preparing to unpack …/29-libruby3.0_3.0.2-7ubuntu2.4_amd64.deb …
Unpacking libruby3.0:amd64 (3.0.2-7ubuntu2.4) …
Selecting previously unselected package libsynctex2:amd64.
Preparing to unpack …/30-libsynctex2_2021.20210626.59705-1ubuntu0.1_amd64.deb
…
Unpacking libsynctex2:amd64 (2021.20210626.59705-1ubuntu0.1) …
Selecting previously unselected package libteckit0:amd64.
Preparing to unpack …/31-libteckit0_2.5.11+ds1-1_amd64.deb …
Unpacking libteckit0:amd64 (2.5.11+ds1-1) …
Selecting previously unselected package libtexlua53:amd64.
Preparing to unpack …/32-libtexlua53_2021.20210626.59705-1ubuntu0.1_amd64.deb
…
Unpacking libtexlua53:amd64 (2021.20210626.59705-1ubuntu0.1) …
Selecting previously unselected package libtexluajit2:amd64.
Preparing to unpack
…/33-libtexluajit2_2021.20210626.59705-1ubuntu0.1_amd64.deb …
Unpacking libtexluajit2:amd64 (2021.20210626.59705-1ubuntu0.1) …
Selecting previously unselected package libzzip-0-13:amd64.
Preparing to unpack …/34-libzzip-0-13_0.13.72+dfsg.1-1.1_amd64.deb …
Unpacking libzzip-0-13:amd64 (0.13.72+dfsg.1-1.1) …
Selecting previously unselected package xfonts-encodings.
Preparing to unpack …/35-xfonts-encodings_1%3a1.0.5-0ubuntu2_all.deb …
Unpacking xfonts-encodings (1:1.0.5-0ubuntu2) …
Selecting previously unselected package xfonts-utils.
Preparing to unpack …/36-xfonts-utils_1%3a7.7+6build2_amd64.deb …
Unpacking xfonts-utils (1:7.7+6build2) …
Selecting previously unselected package lmodern.
Preparing to unpack …/37-lmodern_2.004.5-6.1_all.deb …
Unpacking lmodern (2.004.5-6.1) …
Selecting previously unselected package preview-latex-style.
Preparing to unpack …/38-preview-latex-style_12.2-1ubuntu1_all.deb …
Unpacking preview-latex-style (12.2-1ubuntu1) …
```

```
Selecting previously unselected package t1utils.
Preparing to unpack …/39-t1utils_1.41-4build2_amd64.deb …
Unpacking t1utils (1.41-4build2) …
Selecting previously unselected package teckit.
Preparing to unpack …/40-teckit_2.5.11+ds1-1_amd64.deb …
Unpacking teckit (2.5.11+ds1-1) …
Selecting previously unselected package tex-gyre.
Preparing to unpack …/41-tex-gyre_20180621-3.1_all.deb …
Unpacking tex-gyre (20180621-3.1) …
Selecting previously unselected package texlive-binaries.
Preparing to unpack …/42-texlive-
binaries_2021.20210626.59705-1ubuntu0.1_amd64.deb …
Unpacking texlive-binaries (2021.20210626.59705-1ubuntu0.1) …
Selecting previously unselected package texlive-base.
Preparing to unpack …/43-texlive-base_2021.20220204-1_all.deb …
Unpacking texlive-base (2021.20220204-1) …
Selecting previously unselected package texlive-fonts-recommended.
Preparing to unpack …/44-texlive-fonts-recommended_2021.20220204-1_all.deb …
Unpacking texlive-fonts-recommended (2021.20220204-1) …
Selecting previously unselected package texlive-latex-base.
Preparing to unpack …/45-texlive-latex-base_2021.20220204-1_all.deb …
Unpacking texlive-latex-base (2021.20220204-1) …
Selecting previously unselected package libfontbox-java.
Preparing to unpack …/46-libfontbox-java_1%3a1.8.16-2_all.deb …
Unpacking libfontbox-java (1:1.8.16-2) …
Selecting previously unselected package libpdfbox-java.
Preparing to unpack …/47-libpdfbox-java_1%3a1.8.16-2_all.deb …
Unpacking libpdfbox-java (1:1.8.16-2) …
Selecting previously unselected package texlive-latex-recommended.
Preparing to unpack …/48-texlive-latex-recommended_2021.20220204-1_all.deb …
Unpacking texlive-latex-recommended (2021.20220204-1) …
Selecting previously unselected package texlive-pictures.
Preparing to unpack …/49-texlive-pictures_2021.20220204-1_all.deb …
Unpacking texlive-pictures (2021.20220204-1) …
Selecting previously unselected package texlive-latex-extra.
Preparing to unpack …/50-texlive-latex-extra_2021.20220204-1_all.deb …
Unpacking texlive-latex-extra (2021.20220204-1) …
```

4. Convert to PDF (replace `000000000` with your student ID)

[ ]:
```
%env STUDENT_ID=261072449
!jupyter nbconvert --to pdf "/content/drive/MyDrive/Colab Notebooks/
↪${STUDENT_ID}_Assignment_4.ipynb"
```

5. Download the resulting PDF file. If you are using Chrome, you can do so by running the following code. On other browsers, you can download the PDF using the file manangar on the left of the screen (Navigate to the file > Right Click > Download).

[ ]:
```python
import os
from google.colab import files
files.download(f"/content/drive/MyDrive/Colab Notebooks/{os.
 ↪environ['STUDENT_ID']}_Assignment_4.pdf")
```

6. Verify that your PDF correctly displays your figures and responses.