In this notebook

Many college courses conclude by giving students the opportunity to evaluate the course and the instructor anonymously. In the article "Beauty in the Classroom: Professors' Pulchritude and Putative Pedagogical Productivity" (PDF), authors Daniel Hamermesh and Amy M. Parker suggest (based on a data set of teaching evaluation scores collected at UT Austin) that student evaluation scores can partially be predicted by features unrelated to teaching, such as the physical attractiveness of the instructor.

In this lab, we will use this data to try and predict the average instructor rating with a multiple linear regression.

Attribution

Parts of this lab are based on a lab assignment from the OpenIntro textbook "Introductory Statistics with Randomization and Simulation" that is released under a Creative Commons Attribution-ShareAlike 3.0 Unported license. The book website is at https://www.openintro.org/book/isrs/. You can read a PDF copy of the book for free and watch video lectures associated with the book at that URL. You can also see the lab assignment that this notebook is based on.

Data

The data were gathered from end of semester student evaluations for a large sample of professors from the University of Texas at Austin. In addition, six students looked at a photograph of each professor in the sample, and rated the professors' physical appearance. More specifically:

> Each of the professors' pictures was rated by each of six undergraduate students: Three women and three men, with one of each gender being a lower-division, two upper-division students (to accord with the distribution of classes across the two levels). The raters were told to use a 10 (highest) to 1 rating scale, to concentrate on the physiognomy of the professor in the picture, to make their ratings independent of age, and to keep 5 in mind as an average.

We are using a slightly modified version of the original data set from the published paper. The dataset was released along with the textbook "Data Analysis Using Regression and Multilevel/Hierarchical Models" (Gelman and Hill, 2007).)

Setup

We will start by importing relevant libraries, setting up our notebook, reading in the data, and checking that it was loaded correctly.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn import metrics
from sklearn import model_selection
from sklearn.linear_model import LinearRegression


from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
%matplotlib inline
```

```
url = 'https://www.openintro.org/stat/data/evals.csv'
df = pd.read_csv(url)
df.head()
```

```
df.columns
df.shape
```

```
   score         rank     ethnicity  gender language  age  cls_perc_eval  \
0    4.7  tenure track      minority  female  english   36       55.81395
1    4.1  tenure track      minority  female  english   36       68.80000
2    3.9  tenure track      minority  female  english   36       60.80000
3    4.8  tenure track      minority  female  english   36       62.60163
4    4.6       tenured  not minority    male  english   59       85.00000

   cls_did_eval  cls_students cls_level  ...   cls_credits bty_f1lower  \
0            24            43     upper  ...  multi credit           5
1            86           125     upper  ...  multi credit           5
2            76           125     upper  ...  multi credit           5
3            77           123     upper  ...  multi credit           5
4            17            20     upper  ...  multi credit           4

   bty_f1upper  bty_f2upper  bty_m1lower  bty_m1upper  bty_m2upper  bty_avg  \
0            7            6            2            4            6      5.0
1            7            6            2            4            6      5.0
2            7            6            2            4            6      5.0
3            7            6            2            4            6      5.0
4            4            2            2            3            3      3.0

   pic_outfit pic_color
0  not formal     color
1  not formal     color
2  not formal     color
3  not formal     color
4  not formal     color

[5 rows x 21 columns]
```

```
Index(['score', 'rank', 'ethnicity', 'gender', 'language', 'age',
       'cls_perc_eval', 'cls_did_eval', 'cls_students', 'cls_level',
       'cls_profs', 'cls_credits', 'bty_f1lower', 'bty_f1upper', 'bty_f2upper',
       'bty_m1lower', 'bty_m1upper', 'bty_m2upper', 'bty_avg', 'pic_outfit',
       'pic_color'],
      dtype='object')
```

```
(463, 21)
```

Each row in the data frame represents a different course, and columns represent variables about the courses and professors. The data dictionary is reproduced here from the OpenIntro lab:

variable	description
`score`	average professor evaluation score: (1) very unsatisfactory - (5) excellent.
`rank`	rank of professor: teaching, tenure track, tenured.
`ethnicity`	ethnicity of professor: not minority, minority.
`gender`	gender of professor: female, male.
`language`	language of school where professor received education: english or non-english.
`age`	age of professor.

variable	description
cls_perc_eval	percent of students in class who completed evaluation.
cls_did_eval	number of students in class who completed evaluation.
cls_students	total number of students in class.
cls_level	class level: lower, upper.
cls_profs	number of professors teaching sections in course in sample: single, multiple.
cls_credits	number of credits of class: one credit (lab, PE, etc.), multi credit.
bty_f1lower	beauty rating of professor from lower level female: (1) lowest - (10) highest.
bty_f1upper	beauty rating of professor from upper level female: (1) lowest - (10) highest.
bty_f2upper	beauty rating of professor from second upper level female: (1) lowest - (10) highest.
bty_m1lower	beauty rating of professor from lower level male: (1) lowest - (10) highest.
bty_m1upper	beauty rating of professor from upper level male: (1) lowest - (10) highest.
bty_m2upper	beauty rating of professor from second upper level male: (1) lowest - (10) highest.
bty_avg	average beauty rating of professor.
pic_outfit	outfit of professor in picture: not formal, formal.
pic_color	color of professor's picture: color, black & white.

Explore data

As always, start by exploring the data:

```
df.describe()
```

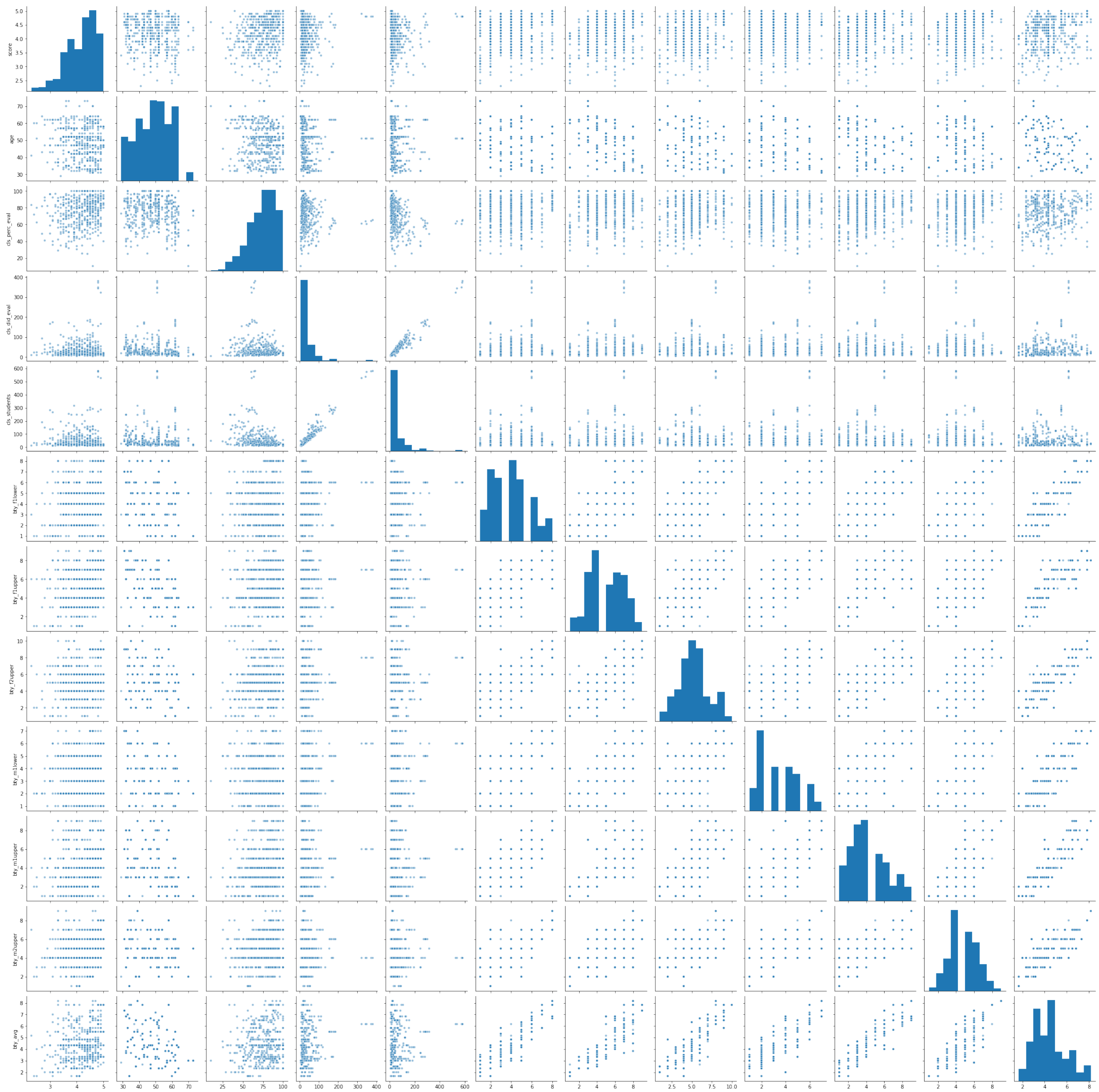
```
            score         age  cls_perc_eval  cls_did_eval  cls_students  \
count  463.000000  463.000000     463.000000    463.000000    463.000000
mean     4.174730   48.365011      74.427788     36.624190     55.177106
std      0.543865    9.802742      16.756311     45.018481     75.072800
min      2.300000   29.000000      10.416670      5.000000      8.000000
25%      3.800000   42.000000      62.696165     15.000000     19.000000
50%      4.300000   48.000000      76.923080     23.000000     29.000000
75%      4.600000   57.000000      87.249170     40.000000     60.000000
max      5.000000   73.000000     100.000000    380.000000    581.000000

       bty_f1lower  bty_f1upper  bty_f2upper  bty_m1lower  bty_m1upper  \
count   463.000000   463.000000   463.000000   463.000000   463.000000
mean      3.963283     5.019438     5.213823     3.412527     4.146868
std       1.873936     1.934437     2.018224     1.637102     2.110586
min       1.000000     1.000000     1.000000     1.000000     1.000000
25%       2.000000     4.000000     4.000000     2.000000     3.000000
50%       4.000000     5.000000     5.000000     3.000000     4.000000
75%       5.000000     7.000000     6.000000     5.000000     5.000000
max       8.000000     9.000000    10.000000     7.000000     9.000000

       bty_m2upper     bty_avg
count   463.000000  463.000000
mean      4.751620    4.417844
std       1.575266    1.527380
min       1.000000    1.667000
25%       4.000000    3.167000
50%       5.000000    4.333000
75%       6.000000    5.500000
max       9.000000    8.167000
```

```python
sns.pairplot(df, plot_kws={'alpha':0.5, 'size': 0.1})
```

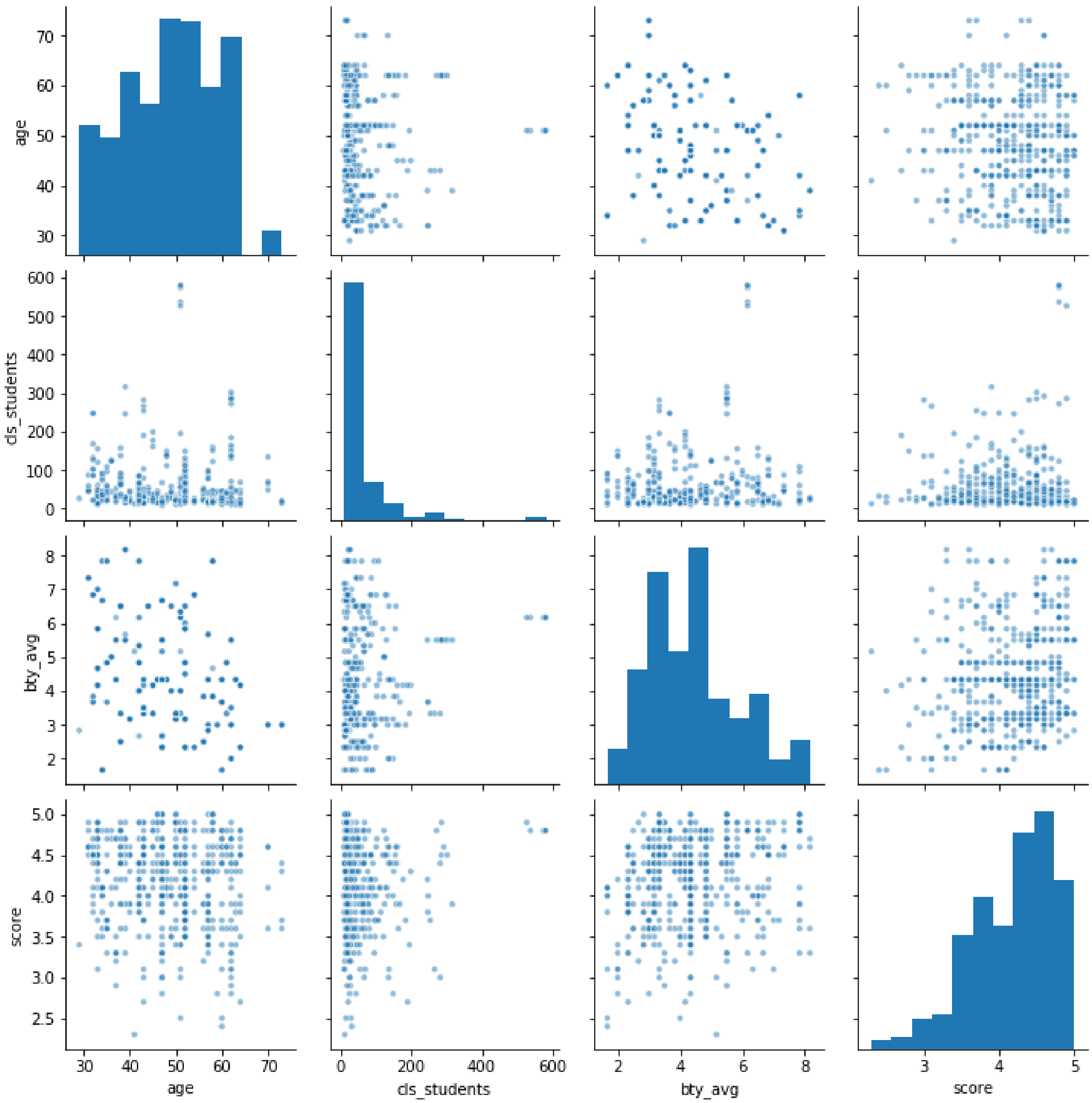
```
<seaborn.axisgrid.PairGrid at 0x7fe2c0c2a5e0>
```



With so many numeric variables, the pair plot is hard to read. We can create a pairplot excluding some variables that we don't expect to be useful for visualization: `cls_perc_eval`, `cls_did_eval`. We will also exclude the individual attractiveness ratings `bty_f1lower`, `bty_f1upper`, `bty_f2upper`, `bty_m1lower`, `bty_m1upper`, `bty_m2upper`, since the overall attractiveness rating is still represented by `bty_avg`.

```python
sns.pairplot(df, vars=['age', 'cls_students', 'bty_avg', 'score'], plot_kws={'alpha':0.5,
    'size': 0.1})
```

```
<seaborn.axisgrid.PairGrid at 0x7fe2a1d47ee0>
```



As part of our exploration of the data, we can also examine the effect of non-numeric variables related to the instructor and the class: `rank`, `ethnicity`, `gender`, `language`, `cls_level`, `cls_profs`, `cls_credits`.

```
for feature in ['rank', 'ethnicity', 'gender', 'language', 'cls_level', 'cls_profs',
    'cls_credits']:
    df.groupby([feature])['score'].describe()
```

```
              count      mean       std  min  25%   50%  75%  max
rank
teaching      102.0  4.284314  0.498263  3.3  3.9  4.40  4.7  5.0
tenure track  108.0  4.154630  0.561104  2.3  3.7  4.35  4.6  4.9
tenured       253.0  4.139130  0.550262  2.4  3.8  4.20  4.6  5.0
```

```
              count      mean       std  min    25%    50%    75%  max
```

```
ethnicity
minority       64.0  4.071875  0.581588  2.7  3.675  4.05  4.525  5.0
not minority  399.0  4.191228  0.536505  2.3  3.850  4.30  4.600  5.0
```

```
        count      mean       std  min  25%  50%  75%  max
gender
female  195.0  4.092821  0.563814  2.3  3.7  4.1  4.5  5.0
male    268.0  4.234328  0.521896  2.4  3.9  4.3  4.6  5.0
```

```
             count      mean       std  min  25%   50%  75%  max
language
english      435.0  4.189655  0.547183  2.3  3.9  4.30  4.6  5.0
non-english   28.0  3.942857  0.434979  3.4  3.6  3.75  4.4  4.8
```

```
           count      mean       std  min  25%  50%  75%  max
cls_level
lower      157.0  4.238217  0.592532  2.5  3.8  4.4  4.7  5.0
upper      306.0  4.142157  0.515104  2.3  3.8  4.2  4.5  5.0
```

```
           count      mean       std  min  25%  50%  75%  max
cls_profs
multiple   306.0  4.184641  0.551177  2.4  3.8  4.3  4.6  5.0
single     157.0  4.155414  0.530529  2.3  3.8  4.3  4.6  5.0
```

```
              count      mean       std  min  25%  50%  75%  max
cls_credits
multi credit  436.0  4.147018  0.542464  2.3  3.8  4.2  4.6  5.0
one credit     27.0  4.622222  0.334357  3.5  4.5  4.7  4.9  5.0
```

Question 1 Describe the relationship between `score` and the overall attractiveness rating `bty_avg`. Is there an apparent correlation? If so, is it a positive or a negative correlation? What about `age` and `cls_students`, do they appear to be correlated with `score`?

Also describe the relationship between `score` and the categorical variables you explored above that are related to characteristics of the *instructor*: `rank`, `ethnicity`, `gender`, `language`. Which of these variables have an apparent correlation with `score`? Is it a positive or a negative correlation?

Are any of the apparent relationships you observed unexpected to you? Explain.

Encoding categorical variables

For *one hot encoding* of categorical variables, we can use the `get_dummies` function in `pandas`. Create a copy of the dataframe with all categorical variables transformed into indicator ("dummy") variables, and save it in a new data frame called `df_enc`. Compare the columns of the `df` data frame versus the `df_enc` data frame.

```
df_enc = pd.get_dummies(df)
df_enc.columns
```

```
Index(['score', 'age', 'cls_perc_eval', 'cls_did_eval', 'cls_students',
       'bty_f1lower', 'bty_f1upper', 'bty_f2upper', 'bty_m1lower',
       'bty_m1upper', 'bty_m2upper', 'bty_avg', 'rank_teaching',
       'rank_tenure track', 'rank_tenured', 'ethnicity_minority',
       'ethnicity_not minority', 'gender_female', 'gender_male',
       'language_english', 'language_non-english', 'cls_level_lower',
       'cls_level_upper', 'cls_profs_multiple', 'cls_profs_single',
       'cls_credits_multi credit', 'cls_credits_one credit',
       'pic_outfit_formal', 'pic_outfit_not formal', 'pic_color_black&white',
       'pic_color_color'],
      dtype='object')
```

Split data

Next, we split the encoded data into a training set (70%) and test set (30%). We will be especially interested in evaluating the model performance on the test set. Since it was not used to train the model parameters (intercept and coefficients), the performance on this data gives us a better idea of how the model may perform on new data.

We'll use the `train_test_split` method in `sklearn`'s `model_selection` module. Since it randomly splits the data, we'll pass a random "state" into the function that makes the split repeatable (same split every time we run this notebook) and ensures that everyone in the class will have exactly the same split.

```
train, test = model_selection.train_test_split(df_enc, test_size=0.3, random_state=9)
# why 9? see https://dilbert.com/strip/2001-10-25
train.shape
test.shape
```

```
(324, 31)
```

```
(139, 31)
```

Simple linear regression

Now we are finally ready to train a regression model.

Since the article is nominally abou the attractiveness of the instructor, we will train the simple linear regression on the `bty_avg` feature.

In the cell that follows, write code to

- use `sklearn` to fit a simple linear regression model on the training set, using `bty_avg` as the feature on which to train. Save your fitted model in a variable `reg_simple`.
- print the intercept and coefficient of the model.
- use `predict` on the fitted model to estimate the evaluation score on the training set, and save this array in `y_pred_train`.
- use `predict` on the fitted model to estimate the evaluation score on the test set, and save this array in `y_pred_test`.

Then run the cell after that one, which will show you the training data, the test data, and your regression line.

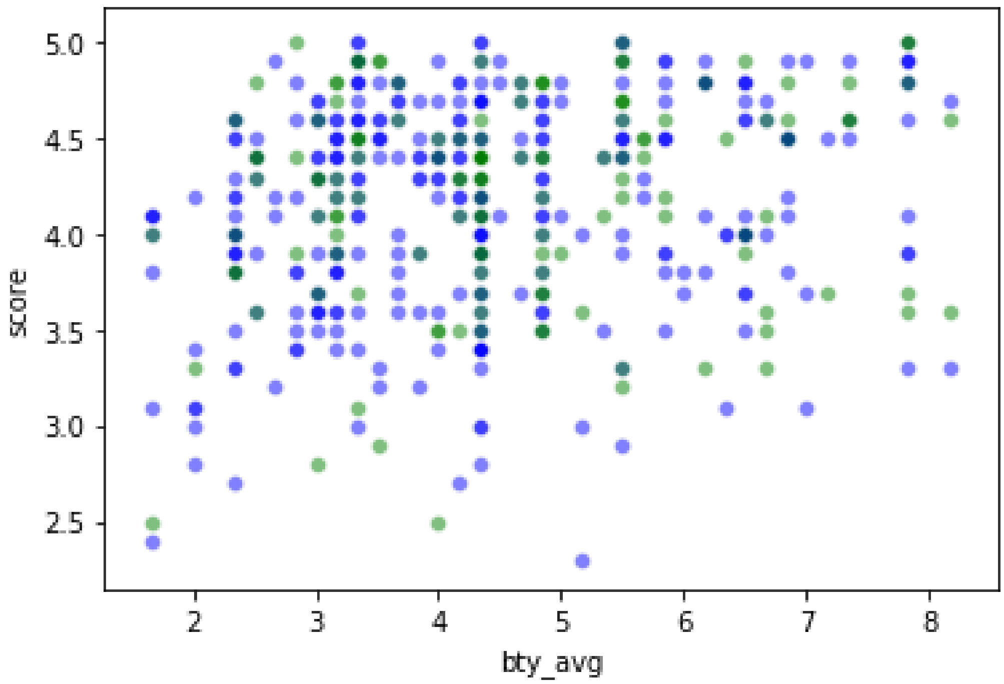
```
# TODO 1
# reg_simple = ...
```

```
# y_pred_train = ...
# y_pred_test = ...
```

```
sns.scatterplot(data=train, x="bty_avg", y="score", color='blue', alpha=0.5);
sns.scatterplot(data=test, x="bty_avg", y="score", color='green', alpha=0.5);
sns.lineplot(data=train, x="bty_avg", y=y_pred_train, color='red');
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-10-12e38b046321> in <module>
      1 sns.scatterplot(data=train, x="bty_avg", y="score", color='blue', alpha=0.5);
      2 sns.scatterplot(data=test, x="bty_avg", y="score", color='green', alpha=0.5);
----> 3 sns.lineplot(data=train, x="bty_avg", y=y_pred_train, color='red');

NameError: name 'y_pred_train' is not defined
```



Evaluate simple linear regression performance

Now we will evaluate our model performance.

In the following cell, write a *function* to compute key performance metrics for your model:

- compute the R2 score on your training data, and print it
- compute the RSS per sample on your training data, and print it
- compute the RSS per sample, divided by the sample variance of `score`, on your training data, and print it. Recall that this metric tells us the ratio of average error of your model to average error of prediction by mean.
- and compute the same three metrics for your test set

```
# TODO 2 fill in the function -
```

```python
def print_regression_performance(y_true_train, y_pred_train, y_true_test, y_pred_test):
    # ...
```

```
  File "<ipython-input-11-8ec32a16a89a>", line 4
    # ...
         ^
SyntaxError: unexpected EOF while parsing
```

Call your function to print the performance of the simple linear regression. Is a simple linear regression on `bty_avg` better than a "dumb" model that predicts the mean value of `score` for all samples?

```python
print_regression_performance(train['score'], y_pred_train, test['score'], y_pred_test)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-12-cbaae82fc60b> in <module>
----> 1 print_regression_performance(train['score'], y_pred_train, test['score'],
    y_pred_test)

NameError: name 'print_regression_performance' is not defined
```

Multiple linear regression

Next, we'll see if we can improve model performance using multiple linear regression, with more features included.

To start, we need to decide which features to use as input to our model. One possible approach is to use every feature in the dataset excluding the target variable, `score`.

You can build and view this list of features by running:

```python
features = df_enc.columns.drop(['score'])
features
```

```
Index(['age', 'cls_perc_eval', 'cls_did_eval', 'cls_students', 'bty_f1lower',
       'bty_f1upper', 'bty_f2upper', 'bty_m1lower', 'bty_m1upper',
       'bty_m2upper', 'bty_avg', 'rank_teaching', 'rank_tenure track',
       'rank_tenured', 'ethnicity_minority', 'ethnicity_not minority',
       'gender_female', 'gender_male', 'language_english',
       'language_non-english', 'cls_level_lower', 'cls_level_upper',
       'cls_profs_multiple', 'cls_profs_single', 'cls_credits_multi credit',
       'cls_credits_one credit', 'pic_outfit_formal', 'pic_outfit_not formal',
       'pic_color_black&white', 'pic_color_color'],
      dtype='object')
```

In the following cell, write code to

- use `sklearn` to fit a linear regression model on the training set, using the `features` array as the list of features to train on. Save your fitted model in a variable `reg_multi`.
- print a table of the features used in the regression and the coefficient assigned to each. If you have saved your fitted regression in a variable named `reg_multi`, you can create and print this table with:

```python
df_coef = pd.DataFrame(data =
                        {'feature': features,
```

```
                         'coefficient': reg_multi.coef_})
df_coef
```

```
# TODO 3
# reg_multi = ...
```

Question 2 Look at the list of features and coefficients, especially those related to the attractiveness ratings.

Are these results surprising, based on the results of the simple linear regression? Explain your answer.

Effect of collinearity

Note especially the coefficients associated with each of the individual attractiveness rankings, and the coefficient associated with the average attractiveness ranking. Each of these features separately seems to have a large effect; however, because they are strongly *collinear*, they cancel one another out.

(You should be able to see the collinearity clearly in the pairplot you created.)

In the following cell, write code to

- create a new `features` array, that drops the *individual* attractiveness rankings in addition to the `score` variable (but do *not* drop the average attractiveness ranking)
- use `sklearn` to fit a linear regression model on the training set, using the new `features` array as the list of features to train on. Save your fitted model in a variable `reg_avgbty`.
- print a table of the features used in the regression and the coefficient assigned to each.

```
# TODO 4
# features = ...
# reg_avgbty = ...
```

Question 3 Given the model parameters you have found, rank the following features from "strongest effect" to "weakest effect" in terms of their effect (on average) on the evaluation score:

- Instructor ethnicity
- Instructor gender

(Note that in general, we cannot use the coefficient to compare the effect of features that have a different range. Both ethnicity and gender are represented by binary one hot-encoded variables.)

Evaluate multiple regression model performance

Evaluate the performance of your `reg_avgbty` model. In the next cell, write code to:

- use the `predict` function on your fitted regression to find $\hat{y}$ for all samples in the *training* set, and save this in an array called `y_pred_train`
- use the `predict` function on your fitted regression to find $\hat{y}$ for all samples in the *test* set, and save this in an array called `y_pred_test`
- call the `print_regression_performance` function you wrote in a previous cell, and print the performance metrics on the training and test set.

```
# TODO 5
# y_pred_train = ...
# y_pred_test = ...
```

Question 4 Based on the analysis above, what portion of the variation in instructor teaching evaluation can be explained by the factors unrelated to teaching performance, such as the physical characteristics of the instructor?

Question 5 Based on the analysis above, is your model better at predicting instructor teaching scores than a "dumb" model that just assigns the mean teaching score to every instructor? Explain.

Question 6 Suppose you are hired by the ECE department to develop a classifer that will identify high-performing faculty, who will then be awarded prizes for their efforts.

Based on the analysis above, do you think it would be fair to use scores on teaching evaluations as an input to your classifier? Explain your answer.

Exploring unexpected correlation

There are some features that we do *not* expect to be correlated with the instructor's score.

For example, consider the "features" related to the photograph used by the students who rated the instructor's attractiveness.

There is no reason that characteristics of an instructor's photograph - whether it was in black and white or color, how the instructor was dressed in the photograph - should influence the ratings of students in the instructor's class. These students most likely did not even see the photograph.

In the next cell, write code to

- create a new `features` array that drops the `score` variable, all of the individual attractiveness rankings, and the variables related to the photograph used for attractiveness rankings.
- use it to fit a model (saved in `reg_nopic`).
- use `reg_nopic` to predict the evaluation scores on both the training and test set
- compute the same set of metrics as above.

```
# TODO 6
# features = ...
# reg_nopic = ...
```

Question 7 Is your model less predictive when features related to the instructor photograph are excluded? Explain.

Finally, we will observe the effect of excluding class-related variables (whether it is an upper-division or lower-division class, number of credits, etc.)

In the next cell, write code to:

- create a new `features` array that drops the `score` variable, all of the individual attractiveness rankings, the variables related to the photograph used for attractiveness rankings, *and* all of the variables that begin with the `cls` prefix.
- use it to fit a model (saved in `reg_nocls`).
- use `reg_nocls` to predict the evaluation scores on both the training and test set
- compute the same set of metrics as above.

```
# TODO 7
# features = ...
# reg_nocls = ...
```

When a machine learning model seems to use a feature that is not expected to be correlated with the target variable (such as the characteristics of the instructor's photograph…), this can sometimes be a signal that information is "leaking" between the training and test set.

In this dataset, each row represents a single course. However, some instructors teach more than one course, and an instructor might get similar evaluation scores on all of the courses he or she teaches. (According to the paper for which this dataset was collected, 94 faculty members taught the 463 courses represented in the dataset, with some faculty members teaching as many as 13 courses.)

For example, consider the output of the following command, which prints all of the one credit courses in the data:

```
df.loc[df['cls_credits']=='one credit']
```

```
     score          rank     ethnicity  gender     language  age  \
124    3.5      teaching  not minority  female      english   52
179    4.4  tenure track      minority  female      english   47
185    4.6  tenure track      minority  female      english   47
245    4.2      teaching  not minority  female      english   50
246    4.7      teaching  not minority  female      english   50
339    4.8  tenure track  not minority    male      english   43
340    4.9  tenure track  not minority    male      english   43
343    4.5  tenure track  not minority    male      english   43
344    4.9  tenure track  not minority    male      english   43
345    4.4  tenure track  not minority    male      english   43
347    4.6      teaching      minority    male      english   50
348    5.0      teaching      minority    male      english   50
349    4.9      teaching      minority    male      english   50
350    4.6      teaching      minority    male      english   50
351    4.8      teaching      minority    male      english   50
352    4.9      teaching      minority    male      english   50
353    4.9      teaching      minority    male      english   50
354    4.9      teaching      minority    male      english   50
355    5.0      teaching      minority    male      english   50
356    4.5      teaching      minority    male      english   50
393    4.8      teaching  not minority    male      english   45
394    4.2      teaching  not minority    male      english   45
396    4.8      teaching  not minority    male      english   45
409    4.7      teaching  not minority  female      english   47
410    4.6      teaching  not minority  female      english   47
411    4.6      teaching  not minority  female      english   47
462    4.1  tenure track      minority  female  non-english   42

     cls_perc_eval  cls_did_eval  cls_students cls_level  ... cls_credits  \
```

```
124       89.47369            17            19     upper  ...  one credit
179      100.00000            16            16     lower  ...  one credit
185       95.23810            20            21     lower  ...  one credit
245       75.00000            24            32     lower  ...  one credit
246       66.66666            14            21     lower  ...  one credit
339       53.57143            15            28     lower  ...  one credit
340       60.00000            18            30     lower  ...  one credit
343       94.44444            17            18     lower  ...  one credit
344       84.61539            22            26     lower  ...  one credit
345       60.00000            18            30     lower  ...  one credit
347       70.83334            17            24     lower  ...  one credit
348       90.90909            20            22     lower  ...  one credit
349       84.00000            21            25     lower  ...  one credit
350       88.46154            23            26     lower  ...  one credit
351       86.36364            19            22     lower  ...  one credit
352       76.92308            20            26     lower  ...  one credit
353       85.00000            17            20     lower  ...  one credit
354       81.81818            18            22     lower  ...  one credit
355       95.23810            20            21     lower  ...  one credit
356       90.47619            19            21     lower  ...  one credit
393       70.58823            12            17     lower  ...  one credit
394       85.00000            17            20     lower  ...  one credit
396       73.07692            19            26     lower  ...  one credit
409       88.23529            15            17     lower  ...  one credit
410      100.00000            10            10     lower  ...  one credit
411       94.11765            16            17     lower  ...  one credit
462       80.00000            28            35     lower  ...  one credit

    bty_f1lower  bty_f1upper  bty_f2upper  bty_m1lower  bty_m1upper  \
124           6            6            4            2            4
179           2            6            6            3            5
185           2            6            6            3            5
245           2            3            5            2            3
246           2            3            5            2            3
339           3            4            4            2            4
340           3            4            4            2            4
343           3            4            4            2            4
344           3            4            4            2            4
345           3            4            4            2            4
347           1            5            4            1            4
348           1            5            4            1            4
349           1            5            4            1            4
350           1            5            4            1            4
351           1            5            4            1            4
352           1            5            4            1            4
353           1            5            4            1            4
354           1            5            4            1            4
355           1            5            4            1            4
356           1            5            4            1            4
393           1            4            2            5            4
394           1            4            2            5            4
396           1            4            2            5            4
409           8            6            6            4            9
410           8            6            6            4            9
```

```
411           8           6           6           4           9
462           3           8           7           4           6

     bty_m2upper  bty_avg  pic_outfit    pic_color
124            7    4.833  not formal        color
179            4    4.333  not formal        color
185            4    4.333  not formal        color
245            4    3.167  not formal        color
246            4    3.167  not formal        color
339            4    3.500  not formal        color
340            4    3.500  not formal        color
343            4    3.500  not formal        color
344            4    3.500  not formal        color
345            4    3.500  not formal        color
347            5    3.333  not formal        color
348            5    3.333  not formal        color
349            5    3.333  not formal        color
350            5    3.333  not formal        color
351            5    3.333  not formal        color
352            5    3.333  not formal        color
353            5    3.333  not formal        color
354            5    3.333  not formal        color
355            5    3.333  not formal        color
356            5    3.333  not formal        color
393            4    3.333  not formal        color
394            4    3.333  not formal        color
396            4    3.333  not formal        color
409            7    6.667  not formal  black&white
410            7    6.667  not formal  black&white
411            7    6.667  not formal  black&white
462            4    5.333  not formal        color

[27 rows x 21 columns]
```

We observe that 10 out of 27 one-credit courses are taught by what seems to be the same instructor - an individual who is a teaching-track professor, minority ethnicity, male, English-language trained, 50 years old, average attractiveness 3.333, and whose photograph is in color and not formal.

This provides a clue regarding the apparent importance of the `cls_credits` variable and other "unexpected" variables in predicting the teaching score. Certain variables may be used by the model to identify the instructor and then learn a relationship between the individual instructor and his or her typical evaluation score, instead of learning a true relationship between the variable and the evaluation score.

To explore this issue further, we will repeat our analysis using two different ways of splitting the dataset:

1. random split
2. random split that ensures that each individual *instructor* is represented in the training data or the test data, but not both.

In the latter case, if the regression model is effectively identifying individual instructors, rather than learning true relationships between instructor/course characteristics and teaching ratings, then the model will perform much worse on the test set for this type of split. This is because the instructors it has "learned" are not present in the test set.

First, we will assign an "instructor ID" to each row in our data frame:

```
instructor_id = df[['rank', 'ethnicity', 'gender', 'language',
```

```
        'pic_outfit', 'pic_color']].agg('-'.join, axis=1)
instructor_id +=  '-' + df['age'].astype(str)
instructor_id +=  '-' + df['bty_avg'].astype(str)

df_enc = df_enc.assign(instructor_id = instructor_id)

df_enc['instructor_id'].head()
```

```
0    tenure track-minority-female-english-not forma...
1    tenure track-minority-female-english-not forma...
2    tenure track-minority-female-english-not forma...
3    tenure track-minority-female-english-not forma...
4    tenured-not minority-male-english-not formal-c...
Name: instructor_id, dtype: object
```

Now we will perform our splits, train a model, and print performance metrics according to the first scheme, in which an instructor may be present in both the training set and the test set.

In the following cell, add code as indicated:

```
ss = model_selection.ShuffleSplit(n_splits=10, test_size=0.3, random_state=9)

for train_idx, test_idx in ss.split(df_enc):
    train = df_enc.iloc[train_idx]
    test = df_enc.iloc[test_idx]

    features = df_enc.columns.drop(['score', 'instructor_id'])
    print('----')

    # TODO 8: add code to train a multiple linear regression using
    # the train dataset and the list of features created above
    # save the fitted model in reg_rndsplit
    # then use the model to create y_pred_train and y_pred_test,
    # the model predictions on the training set and test set.
    # Finally, use print_regression_performance to see the
    # model performance
    #
    # reg_rndsplit = ...
    # y_pred_train = ...
    # y_pred_test = ...


    print_regression_performance(train['score'], y_pred_train, test['score'], y_pred_test)
```

```
----
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-21-6f1aae6d8ea4> in <module>
     21
     22
---> 23     print_regression_performance(train['score'], y_pred_train, test['score'],
    y_pred_test)
```

```
NameError: name 'print_regression_performance' is not defined
```

Then, we will perform our splits, train a model, and print performance metrics according to the second scheme, in which an instructor may be present in either the training set or the test set, but not both.

In the following cell, add code as indicated:

```
gss = model_selection.GroupShuffleSplit(n_splits=10, test_size=0.3, random_state=9)
for train_idx, test_idx in gss.split(df_enc, groups=instructor_id):
    train = df_enc.iloc[train_idx]
    test = df_enc.iloc[test_idx]

    features = df_enc.columns.drop(['score', 'instructor_id'])

    # TODO 9: add code to train a multiple linear regression using
    # the train dataset and the list of features created above
    # save the fitted model in reg_grpsplit
    # then use the model to create y_pred_train and y_pred_test
    # the model predictions on the training set and test set.
    # Finally, use print_regression_performance to see the
    # model performance
    #
    # reg_grpsplit = ...
    # y_pred_train = ...
    # y_pred_test  = ...


    print_regression_performance(train['score'], y_pred_train, test['score'], y_pred_test)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-22-3fc874dd4457> in <module>
     19
     20
---> 21     print_regression_performance(train['score'], y_pred_train, test['score'],
    y_pred_test)

NameError: name 'print_regression_performance' is not defined
```

Question 8 Based on your analysis above, do you think your model will be useful to predict the teaching evaluation scores of a new faculty member at UT Austin, based on his or her physical characteristics and the characteristics of the course?

Data leakage

In this case study, we saw evidence of data leakage: The identity of the instructor "leaked" into the data set, and then the model learned the instructor ID, not a true relationship between instructor characteristics and teaching evaluation scores.

As a result, the model had overly optimistic error on the test set. The model appeared to generalize to new, unseen, data, but in fact would not generalize to different instructors.

Another example of data leakage:

You have been hired to develop a new spam classifier for NYU Tandon email. To collect a dataset for the spam classification task, you get 5,000 NYU Tandon students, faculty, and staff who agree to manually label every email they receive for the week of March 15-March 21 as "spam" (about 5%) or "not spam" (about 95%). They then share all the emails and labels with you. For example, here are a few of the emails you got from Volunteer 1, who is a member of the ECE department:

Subject	From	To	Label
April Funding Opportunities	Office of Sponsored Programs	All Tandon Faculty	Not Spam
ML TA meeting next week	Student 23451 (redacted)	Fraida Fund	Not Spam
Pass/fail option for students this semester	Ivan Selesnick	ECE Faculty	Not Spam
A question about quiz1	Student 19245 (redacted)	Fraida Fund	Not Spam
Re: your account is locked	PayPall	Fraida Fund	Spam
Fwd: Gradescope Webinar	Ivan Selesnick	ECE Faculty	Not Spam
Memo to Faculty on COVID-19 Protocols	Provost Katherine Fleming	All Tandon Faculty	Not Spam

You assign the emails from volunteers 1-2,500 to a training set and use it to fit a classifier, then compute the classifier accuracy on the emails from volunteers 2,501-5,000.

Your classifier does really well on the emails from volunteers 2,501-5,000 - in fact, it is 99.9999% accurate! But when you deploy it in production, it misses a lot of spam. Based on the description above, what mistake did you make that caused your performance estimate to be overly optimistic? How would you fix it?

Also potential for data leakage when:

- Learning from adjacent temporal data
- Learning from data that includes duplicate
- Learning from a feature that is a proxy for target variable

How can we detect data leakage?

- Surprising patterns in data (via exploratory data analysis)
- Performance is "too good to be true"
- Features that shouldn't be important (based on common sense/domain knowledge) play a role in reducing error
- Early testing in production