
0.0.1 Question 1c

Linear models are sensitive to multicollinearity among the columns of the design matrix. So, let's determine the extent of multicollinearity in the college rankings dataset.

In the following cell, we provide you with `arranged_cleaned_college_data`, which is a copy of `cleaned_college_data` containing just a subset of the columns arranged in a particular order.

Create a visualization that shows the pairwise correlation between each combination of columns in `arranged_cleaned_college_data`.

- For 2-D visualizations, consider the `sns.heatmap()` documentation.
- For full credit, title your plot, and set `annot=True`. This makes the plot easier to interpret.
- You may find your plot easier to read with a different color scale. For example, try including `cmap="coolwarm"` inside of `sns.heatmap()`.

Hint: Your plot should show 10×10 values corresponding to the pairwise correlations of the selected columns in `arranged_cleaned_college_data`:

```
['Overall Score (0-100)',
'Peer Assessment Score (1-5)',
'Predicted 6yr graduation rate',
'Actual 6yr graduation rate',
'Graduation and retention rank',
'Student Excellence rank',
'Acceptance rate',
'Financial resources rank',
'Is Public',
'Is Private']
```

```
In [10]: # Running this cell helps you get a subset of cleaned_college_data with the above columns
         arranged_cleaned_college_data = cleaned_college_data[
             ['Overall Score (0-100)',
              'Peer Assessment Score (1-5)',
              'Predicted 6yr graduation rate',
              'Actual 6yr graduation rate',
              'Graduation and retention rank',
              'Student Excellence rank',
              'Acceptance rate',
              'Financial resources rank',
              'Is Public',
```

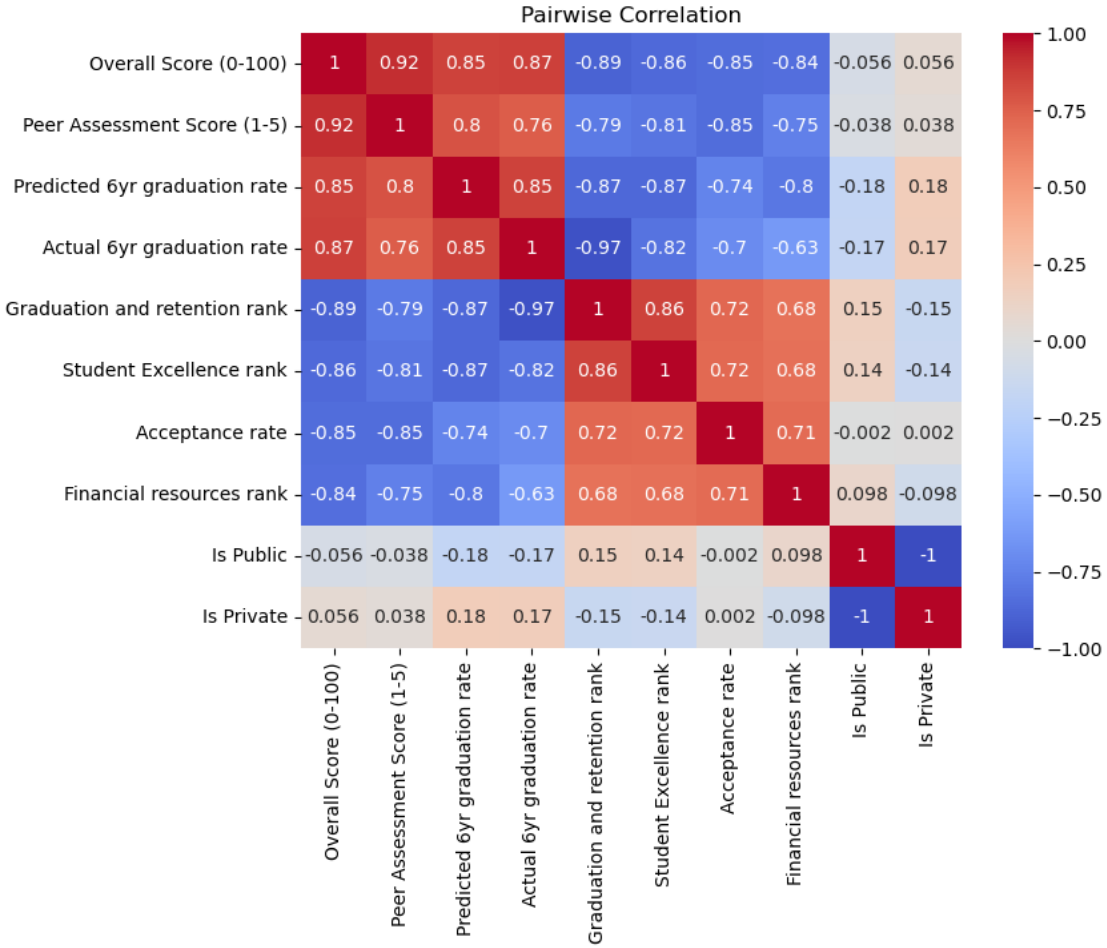
```
        'Is Private']
]
```

Your output figure should look similar to the following example:

```
In [11]: fig, ax = plt.subplots(figsize=(8, 6))

         pairwise_corr = arranged_cleaned_college_data.corr(method='pearson')
         sns.heatmap(data=pairwise_corr, cmap='coolwarm', annot=True)

         plt.title('Pairwise Correlation');
```



0.0.2 Question 1d

Do you notice any patterns in the plot from part (c)? What might explain these patterns? Comment and hypothesize on at least two patterns you notice.

Here are some example questions to ponder:

1. Why do some feature-pairs have correlations of ± 1? Is this a problem?
2. What does the correlation between pairs of features (i.e., graduation-related statistics) look like? Is the magnitude of any of the correlations problemtatically close to 1?
3. Are any features particularly strong predictors of the outcome? If so, why do you think this is the case?
4. Do any features seem potentially redundant? In other words, do you suspect that any features provide similar information about the outcome as other features?

Some feature-pairs have a correlation of 1 because they represent the relationship of a feature with itself. This is expected since any feature will naturally have a perfect positive correlation with itself, as their values are identical across all observations. Some feature-pairs have a correlation of -1 because it is a perfectly inverse relationship. For example, `Is Public` and `Is Private` have a correlation of -1 because a college can only be either public or private, but not both. Knowing the value of one automatically tells you the other, which makes one of these features redundant in the model and introduces multicollinearity, as they are linear combinations of each other. This is an issue because it prevents the model from computing a unique solution, where the design matrix cannot be full rank.

0.0.3 Question 1e

If we tried to fit a linear regression model with an intercept term using all features in `cleaned_college_data`, we might run into some problems when fitting our model. The Data 100 staff suggests that we perform the following operation to the `DataFrame` before fitting a model:

```
In [12]: # You must run this cell to achieve pass the public tests for later questions.
         cleaned_college_data = cleaned_college_data.drop('Is Private', axis=1)
```

Describe the reasoning behind this operation. What problem(s) do we avoid by removing the `Is Private` column from the model fitting process?

By removing the `Is Private` column from the model fitting process, we prevent multicollinearity caused by linear dependence between `Is Public` and `Is Private`. Since one is the direct inverse of the other, keeping both would make the design matrix not full rank, preventing the model from computing a unique solution. So, removing the `Is Private` column ensures that our model can be properly fitted and avoid issues related with multicollinearity.

0.0.4 Question 2b

Let's visualize the model performance from part 2(a). Plot the following: 1. The observed values vs. the predicted values on the test set. 2. The residuals plot. Recall that for multiple linear regression, we plot the residuals against the predicted values.

In both plots, the predicted values should be on the x-axis.

Note: * For a full-credit solution, you should use `plt.subplot()` (documentation) so that you can view both visualizations side-by-side. * The method `plt.subplot({# of rows}{# of cols}{index of plot})` sets the plottable area to the `index` of a `# rows` by `# cols` grid. * For example, `plt.subplot(121)` sets the plottable area to the first index of a 1x2 plot grid. Calling `Matplotlib` and `Seaborn` functions will plot on the first index. When you're ready to start plotting on the second index, run `plt.subplot(122)`. * **Remember to add a guiding line to both plots where $\hat{Y} = Y$, i.e., where the residual is 0**. * `plt.plot()`(documentation) and `plt.axhline()`(documentation) might be helpful here! * Make sure to add descriptive titles and axis labels. * To avoid distorted aspect ratios, ensure the limits of the x-axes for both plots are the same.

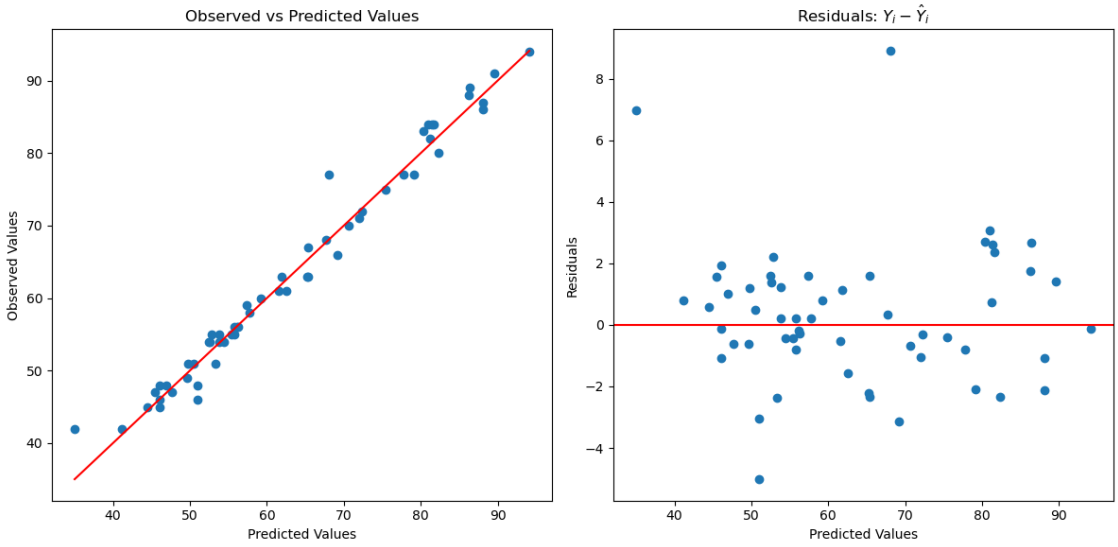
```
In [15]: plt.figure(figsize=(12,6))      # do not change this line
         plt.subplot(121)                # do not change this line
         # 1. plot observations vs. predictions
         plt.scatter(Y_test_pred, Y_test)
         plt.plot([min(Y_test_pred), max(Y_test_pred)], [min(Y_test_pred), max(Y_test_pred)], color='re

         plt.title('Observed vs Predicted Values')
         plt.xlabel('Predicted Values')
         plt.ylabel('Observed Values')

         plt.subplot(122)               # do not change this line
         # 2. plot residual plot
         residuals = Y_test - Y_test_pred
         plt.scatter(Y_test_pred, residuals)
         plt.axhline(0, color='red')

         plt.title('Residuals: $Y_i - \hat{Y}_i$')
         plt.xlabel('Predicted Values')
         plt.ylabel('Residuals')

         plt.tight_layout()
```


Observed vs Predicted Values
Observed Values
Predicted Values
40
50
60
70
80
90
Residuals: $Y_i - \hat{Y}_i$
Residuals
Predicted Values
−4
−2
0
2
4
6
8

0.0.5 Question 2c

Describe what the plots in part (b) indicate about this linear model. In particular, are the predictions good, and do the residuals appear uncorrelated with the predictions?

The Observed vs. Predicted Values plot shows that the predicted values generally follow the observed values along the red line, indicating that the model captures the overall trend in the data. There is some spread around the line, suggesting that the predictions are not perfect and have some variability. Points that deviate noticeably from the red line may represent outliers or cases where the model's predictions are less accurate, suggesting that the model struggles with these specific observations.

The Residuals plot shows that the residuals are centered around zero and are randomly scattered without any specific pattern or trend. This shapeless cloud of points indicates that the residuals are uncorrelated with the predicted values, which is what we want in a linear regression model. These plots show that the linear model does a reasonable job capturing the relationship between the variables. While there are some larger differences between the observed and predicted values, the residuals suggest that the model is fitted well with no obvious pattern in the errors.

Question 3d(i) Let us first interpret **Model B**, the linear regression model that use a subset of features from our dataset.

```
In [ ]: display(Markdown('#### Model B: Subset of Features'))
        print_confidence_intervals(partial_feature_models, partial_feature_cis)
```

Are θ_1 and θ_2 significantly different than 0? How do you know?

Does your answer imply that the relationship between `Overall Score (0-100)`, `Peer Assessment Score (1-5)`, and `Acceptance rate` are causal? Do you think the relationships are causal? Explain.

θ_1 and θ_2 are significantly different than 0, as the features' lower and upper confidence interval bounds do not contain 0. The relationship between `Overall Score (0-100)`, `Peer Assessment Score (1-5)`, and `Acceptance rate` are not causal. While it is true that there may be associations between these features, we cannot say correlation implies causation. There may be other underlying factors that could be influencing the observed relationships.

Question 3d(ii) In what situation(s) would you prefer a more compact model with just key features, like Model B? On the other hand, in what situation(s) would you want to consider many features, like in Model A? Explain your answer to both of these questions.

You would prefer a model with just key features like Model B when you want to reduce complexity and lower the risk of overfitting, especially when working with a smaller dataset. This makes it easier to interpret the relationship between features and the target variable. However, if the model is too simple, it may fail to capture important relationships in the data. So, you would prefer a model with many features like Model A when you have a large dataset and want to capture more complex patterns. Including more features can improve predictive performance, as long as there is enough data to support them and prevent overfitting.

0.0.6 Question 4b

Using the `simulate` function from above, we can compute the model risk, model variance, and variance-to-risk ratio of **Model B**:

```
In [ ]: x_last = X.iloc[X.shape[0] - 100]
        y_last = Y.iloc[X.shape[0] - 100]

        (
            partial_feature_model_risk,
            partial_feature_model_var,
            partial_feature_model_ratio
        ) = simulate(x_last[model_b_features], y_last, partial_feature_models)

        print('Model B risk:')
        print(partial_feature_model_risk)

        print('Model B variance:')
        print(partial_feature_model_var)

        print('Model B ratio:')
        print(partial_feature_model_ratio)
```

Comment on the variance-to-risk ratio for Model B (`partial_feature_ratio`).

- Does the model variance appear to be the dominant term in the bias-variance decomposition? If not, what term(s) dominate the bias-variance decomposition?

Then, given your conclusion above, describe what operation(s) you might perform to reduce the model risk.

The model variance does not appear to be the dominant term in the bias-variance decomposition, as it only accounts for approximately 10% of the total model risk. This suggests that either the model bias or observational bias is the dominant term. Since the observational variance cannot be reduced, I should focus on reducing model bias. This could involve adding new features, transforming existing features, or using a more flexible model that allow for a better fit to the data. I could also use regularization techniques like Lasso and Ridge to manage the complexity of the model. These can help reduce variance and prevent overfitting by penalizing large coefficients. This would find the right balance between bias and variance, minimizing the total model risk and improving the model's ability to generalize to unseen data.