Data Visualization [Optional Review Questions]:

Describe how you could modify a scatter plot to encode information about a categorical variable.

A scatter plot plots two (or three) quantitative variables. One way to encode additional information about a categorical variable is to use coloring. By assigning each unique label in the categorical variable a color and using it to color the corresponding points on the scatter plot, you can have the scatter plot display both categorical and quantitative data.

List two ways you can solve overplotting.

1. Sampling a subset of points to graph
2. Using smoothing techniques (eg. hex plots in scatter plots, larger bin sizes in histograms)

Standardization and Normalization:

What is the difference between standardization and normalization?

Standardization changes the dataset to have a mean of 0 and a standard deviation of 1 (note that this is not a unit gaussian; if the original distribution is not normal, the standardized distribution will not be either). Normalization changes the dataset to be between the range 0-1. An easy way to distinguish the two is that standardization zero-means the data, while normalization does not.

Given a numpy array `x` that contains numerical values, write a line of code that *standardizes* `x`.

```
x = (x - np.mean(x)) / np.std(x)
```

CIFAR10 has 50000 32x32 pixel images, where each pixel is comprised of three channels for the RGB values. Say the dataset is stored as a 50000 x 32 x 32 x 3 tensor. Let's say we want to take the channelwise mean and standard deviation. Across which axes will we do this?

Channelwise mean means we want the mean for each RGB channel, so our mean should be a 3 tensor. Then, we can take the mean across axes (0, 1, 2) and this should return a 3 tensor which is our channelwise mean.

One-Hot Encoding:

Suppose you have a large dataset containing survey data, and one of the features is the participant's hometown. Assume that there are a very large number of unique towns. Would it be appropriate to directly apply one-hot encoding on the feature in this situation?

This would not be an appropriate situation to use one-hot encoding in, since one-hot encoding will produce an excessively wide data matrix. Additionally, because the number of unique towns is very large, many features will be sparse or potentially even degenerate into basis vectors, as participant responses are widely spread out. Instead of directly applying one-hot encoding, it might be useful to first group the towns by a larger geographic area such as counties before encoding. Furthermore, regions that only a handful of people responded with can be grouped together as an "Other" class. Applying one-hot encoding to this preprocessed variable will yield a more reasonable result.

Now assume this data is given to you in a pandas dataframe called `survey` with a "hometown" feature. After appropriately preprocessing the feature, we want to one-hot encode it. Write a code snippet that replaces the original hometown feature with one-hot encoded features.

```
encoded_features = pd.get_dummies(survey.hometown)
survey.drop("hometown", axis = 1, inplace = True)
survey = pd.concat([survey, encoded_features], axis = 1)
```

Processing Text Fields:

Does the following pattern fully match with the text? Ignore the quotes, they just denote that they're both strings.

- Pattern: "\w+@\w*"
- Text: "eecs@berkeley.edu"

The pattern does not match the text because the period in "berkeley.edu" does not match the trailing "\w*" after the "@" symbol. The "\w" meta character only matches alphabetic characters, numeric characters, and underscore. This means that only "eecs@berkeley" will match while the trailing ".edu" does not match.

Given that you have records that will always have the same format as the following log:

```
127.0.0.1 user-identifier frank [10/Oct/2000:13:55:36 -0700] "GET
/apache_pb.gif HTTP/1.0" 200 2326
```

Write a line of regex that extracts the date, month, and year in separate capture groups. Assume that the record is stored in a string called `log` and `re` is already imported.

```
re.findall(r"\[(\d+)\/(\w+)\/(\d+)", log)
```

Handling Imbalanced Class Distributions

Describe one disadvantage of directly modifying the dataset to balance classes instead of using an alternative method such as changing the evaluation metric or collecting more data.

One disadvantage of modifying the dataset directly with a resampling method is that it fundamentally changes the reality of the dataset. If the class imbalance in the dataset accurately reflects the class imbalance in the real world, then we are providing the model with inaccurate information. This means that we must always be careful of what the intention is when using a resampling method.

In what kinds of data/problems is SMOTE a good algorithm for generating synthetic samples? When would it not be appropriate?

SMOTE is good in problems where we can expect data points that are close to each other to have the same class. This usually means that the data is ideal for some kind of clustering algorithm, where the data points are close to one another. It is not appropriate when this assumption breaks down. For example, if the minority class has some convex curve in the distribution, we would not want to put data points above the boundary due to the convex hull. Or if the data is in a ring, we wouldn't want to put samples inside of the circle. This can always be mitigated with more dense data, but a lack of data is usually why we use SMOTE in the first place.

Handling Null/Missing Values:

Briefly outline the two overarching methods that are used to handle null values.

The two general classes of methods are deletion and imputation. Deletion is the act of deleting the offending data points or features that contain the null values. Imputation is the act of filling the null values with some synthetic data. There are many different methods to perform imputation, such as with a summary statistic or training a model. One special case that doesn't fall under either method is that nulls can be treated as their own label for categorical data and simply left as is.

Given a pandas dataframe `housing` with a feature "price", write a line of code that replaces null values with the mean price. Which one of the overarching methods is this?

```
housing.fillna({"price": housing.price.mean()}, inplace = True)
```

This method is imputation, where null values are being filled with a summary statistic.

When using a model such as linear regression or k nearest neighbors to impute null values, what is one concern to always look out for?

One should always be wary of the potential for these models to overfit to the data, especially if the dataset is relatively small.

Removing Outliers (without OMP):

What is a scenario where you wouldn't want to remove outliers?

In a situation where your model is very critical and must work under ALL conditions (eg. building an airplane engine and studying data about temperature vs. engine failure), then removing outliers is not appropriate.

What is the main problem shared by all outlier removal methods we've provided?

All methods we've listed, especially outlier removal by "common sense" violations and visual separation on graphs, are prone to introducing subjective bias in the process. This is because they require humans to make best judgement calls, which may vary from person to person.

Suppose you have a pandas dataframe `titanic` that contains the Titanic dataset. After plotting passenger fares on a histogram, you decide to remove all rows where the fare feature is larger than 200. Write a line of code that performs this operation.

```
titanic = titanic.loc[titanic.fare < 200]
```

Image Data Augmentation:

Why is it important to only apply transforms to the training set and not the validation set?

By applying transforms to the dataset, we are artificially increasing the complexity of the dataset. The motivation to doing this to the training set is to make ourselves more robust to the validation set. If we apply the same transforms to the validation set, we are now effectively training for the exact same problem that we know we will see in the validation set, so instead of gaining the ability to generalize to new variations, we have added those variations into the base set and we can't consider them variations anymore.

Let’s say we decide to take five random crops (randomly picking between the set of corner crops and the center crop) and then a random horizontal flip as our augmenting scheme. By what factor are we increasing the effective size of the dataset?
For each image, we have five possible crops. Each crop can either be the original crop or a horizontally flipped variant. Thus, we have ten possibilities for each image, so our dataset increases by a factor of ten.

Data Decorrelation and Whitening
Why do we decorrelate features?
Any of the following reasons are valid: To reduce noise in true features, to improve the convergence of optimization algorithms, some algorithms assume that features are implicitly independent of each other, etc.

What is the intuition behind using the U matrix as our linear transformation in both PCA whitening and ZCA whitening?
The U matrix contains the column span of the matrix A. Thus, by using the U matrix in our transformation, we are implicitly transforming our raw data to the vector space span by the columns of our data. Since our columns are probably linearly independent (with enough data), we expect that these new features will be non-correlated.