1. Given a data set and two models, how can you be sure which model is better? Explain what you will do to assess which one is better?

To determine the best model, we need to use different criteria to evaluate our two models. There are different ways to evaluate and compare models; thus, it is critical to use the most efficient and accurate ways to evaluate the models. Here are some assessments that can be used to measure the fit of the model: R square, boxplot, scatterplot, residual plot, evaluating model methods, AIC and BIC. However, not all assessments give us the "right" model. It is important to pick the right assessment based on the dataset and models.

The first method that comes into my mind is to use R squared. However, R squared is not a wise choice to evaluate our model in my point of view. The reasons are the R square cannot measure how good our fit is. The value of R square could be arbitrarily different from the real performance of our models. Also, R squares tell us nothing about the prediction error. It is better to use Mean square error to measure prediction error. Our goal is to find the best model for future predictions. We want our model to accurately predict Y for its X.

First, I will compute the MSPE on the two models as one of the measurements to determine which model is best. I would use for loop and apply Cross-Validation to sampling the model on measuring model error. In each split, I will rescale MSPEs relative to best and make a boxplot for the RMSPEs. I would evaluate the two models based on the MSPE and relative-RMSPE boxplot plot. By using a boxplot, we could see the outliers, intervals, and mean for each model. I will compare the MSPEs between the models to see how well the model predicts the true value, I will select the model with the smallest MSPE.

We know that in the given dataset, not all variables are important. Some variables can represent the others and some variables can cause bias to the model. If we want to know which variables are important, then we can use the information criteria where it combines the closeness of a fit with the complexity of a model. Thus, there are another two statistical approaches that can be used to estimate how well a given model fits a dataset and how complex the model is. They are AKAIKE INFORMATION CRITERION and BAYESIAN INFORMATION CRITERION. The first component of the AIC, nlog(sMSE) which gets smaller as the model fits data more closely. The more parameters in the model will lead to larger n and larger first components. The second component 'wk' gets larger as the model gets more complex. We want to select the AIC with the minimum value, which will be the best model. AIC will select the best model that explains the largest amount of variation with fewer explanatory variables. In our case, we can compare the value of AIC on both models and select the model with lower AIC. If two model explains the same amount of variation, then AIC will choose the model with fewer parameters. The difference between the AIC and the BIC is the greater penalty imposed on the number of parameters. With BIC, we will also

prefer the model with lower BIC values. BIC will penalize the model more for its complexity compared to AIC. The more complex model will be less likely selected by BIC. This also causes the BIC to choose the model that is too simple. In this case, the model chosen by BIC is never larger than the model chosen AIC. I would perform a 10-folds Cross-validation on both models and in each fold, I will calculate the AIC and BIC values for each model. Then I will select the model with minimum AIC and BIC values. If different models are selected by AIC and BIC, I would prefer to use the model that is selected by AIC. Since AIC penalizes complex models less and more focused on the model performance.

In conclusion, I will use MSPEs, AIC, and BIC with a combination of 10-folds cross-validation to assess my two models. I will pick the model based on those three assessment values.

2. Describe what a principal component analysis does. What are the key steps of performing PCA?

To understand PCA, we first need to know what is eigenvalues and eigenvectors. They are the core of building a PCA. The eigenvectors determine the directions of the space, and the eigenvalues determine the magnitude. With a given dataset or model, we know that not all variables are useful and some variables can cause more bias than the others. To reduce variance in prediction, we need to reduce the number of parameters to be estimated. Sometimes explanatory variables are highly correlated. Thus, it would be helpful to have a way to reduce the correlated variables to a smaller number that contains the most information about the datasets. PCA is a widely used dimensional-reduction method that reduces the dimension of large datasets. This method reduces the dimension by dropping less important variables while keeping as much information as possible.

There is also some other component that is involved in PCA that we need to know before looking into the key steps of PCA. To understand how PCA works, we should know the following: mean, variance, covariance, and standard deviation. Mean is the average value of all x's in the set X, which is the sum of all data points divided by the number of data points, n. The standard deviation is the square root of the average square distance of data points to the mean. Variance is the measure of the data's spread. Covariance measures the directional relationship between two random variables. In summary, PCA's goal is to increase the computational efficiency by reducing the dimensions of a d-dimensional dataset by projecting it onto a k-dimensional subspace while retaining most of the information of the dataset.

PCA can be broken down into 3 key steps. One of the most important steps is to standardize the range of continuous variables. If there are large differences between the ranges of initial variables, the variables with larger ranges will affect those with small ranges, which will lead to bias results. Thus, it is critical to perform standardization before PC. Then, we need to find the direction within X that explains the most variability and choose the line where the variance is highest. The second most important step in PCA is to compute eigenvectors and eigenvalues. We know that eigenvectors and eigenvalues are critical to PCA. In this step, we need to know if there are any relationships between the variables. We need to identify the variables that are highly correlated and compute the covariance matrix. We need to compute the eigenvectors and eigenvalues from the covariance matrix to

determine the principal components of the data. The eigenvectors of the covariance matrix are the directions of the axes where there is the most variance. Eigenvalues are the coefficients of eigenvectors and give the variance carried in each principal component. We rank the eigenvectors based on their eigenvalues, from highest to lowest. Then, we will get the principal components based on the significance level.

Finally, the last key step I think is to select the principal components. Since we hope that to use a few principal components to explain most of the variability in X. Sometimes parameters are too large to work with or the information it contains is more than we need. We know that principle components are a normalized linear combination of the original predictors and the first components contain most information of the datasets. Thus, the PCA tries to put the maximum information into the first component. The first principle component determines the direction of the highest variability in the data. The second and first principal components are uncorrelated and their direction is orthogonal. The second principal component contains the remaining variance in the dataset. The variance of points around the second principal component is less than the variance around the first principal component. We keep repeating this until there is no more variability. We hope to use a few ($M<p$) PCs to explain most of the variability in X. Thus, what we choose for M can be significant for our PCA performance.

3. In the neural network, what's the intuition behind the bias term in a neuron?

Speaking about Neural networks, the first thing that comes into my mind is artificial intelligence. The fundamental concept of a neural network comes from artificial intelligence and it becomes a popular prediction tool not only in the statistical field. The neural network is known for adapting to changing input and it can generate the best result without redesigning the output criteria. A neural network contains layers of interconnected nodes. So how is a neural network built? A neural net is constructed with a set of input X "feds forward" into hidden layers of hidden nodes. A node in a neural network is a function of a linear combination of inputs. Nodes are also called "hidden nodes" because they are neither measurable for their inputs nor output. They can be iterated to feed-forward into one or more hidden layers. Then it combines the results of the final hidden layer into a prediction. The closer the approximations means more hidden nodes. Thus, neural networks can be excellent predictors.

So how exactly NN works? Let's assume that there is only one response variable Y being predicted. Then we create hidden nodes Z1...ZM from inputs X1….Xp such that there are p +1 parameters per hidden node. Each Zm is constructed with an activation function where is a combination of weights. Where the bias is a constant that helps the model to fit in the best way possible. The activation function is a nonlinear transformation that is used to the input before sending it to the next layer of neuron and is the same for all m. After passing

through every hidden layer, we move to the last layer which is the output layer. The data get passed to the input layer and we receive output in the output layer.

With some basic understanding of neural networks, we know that in every hidden layer we add a bias term to the function. What is the purpose behind the bias term? A bias term is just a simple constant term but it can also be powerful which helps the model to learn better on the given data. We know that an activation function takes input x and is multiplied by a weight ‘w'. Adding bias term allows us to shift the activation function by a constant amount to left or right on an x-axis. In a simple case without bias term, a single input neural is directly connected with an output neuron. The output is calculated by activation function where the input x multiple by a weight. The changing of weight value will only alter the steepness of the function. The model will train over points that pass through the origin only and the fit might be performing poorly. This also does not apply to a real-world situation. Thus, adding a bias term allows us to shift the function horizontally that not only passes through the origin. It also helps increase the flexibility and complexity of the model. Adding a bias term can help us get better prediction results.

.