**Project-Based Course Overview**

Welcome to Statistical Data Visualization with Seaborn! This is a project-based course which should take approximately 1.5 hours to complete. Before diving into the project, please take a look at the course objectives and structure:

**Course Objectives**

In this course, we are going to focus on three learning objectives:

1. Produce and customize various chart types with Seaborn.
2. Apply feature selection and feature extraction methods with scikit-learn.
3. Build a boosted decision tree classifier with XGBoost.

By the end of this course, you will be able to generate publication-quality graphs using Seaborn and Python to analyze data.

**Course Structure**

This course is divided into 4 parts:

1. Course Overview: This introductory reading material.
2. Statistical Data Visualization with Seaborn Project: This is the hands on project that we will work on in Rhyme.
3. Graded Quiz: This is the final assignment that you need to pass in order to successfully complete the course and earn a Course Certificate.

**Project Structure**

The hands on project on Statistical Data Visualization with Seaborn is divided into the following tasks:

**Task 1: Importing Libraries and Data**

* In this task, we will briefly recap our exploratory data analysis of the Breast Cancer Wisconsin (Diagnostic) Data Set in the previous project.

**Task 2: Dropping Correlated Columns from Feature List**

* Using the heatmap of the correlation matrix, we were able to identify columns to be dropped.
* Out of a set of correlated features, we will preserve the one that best separates the data.
* We will identify these salient features using the violin plots and swarm plots produced in the previous project.

**Task 3: Classification using XGBoost (minimal feature selection)**

* Drop 15 columns that are correlated out of a total of 33 columns.
* To verify that there are no remaining correlated features, plot another correlation matrix and inspect the Pearson correlation coefficient.
* Next, use a [helper function](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html) from scikit-learn to create split the data into training and test sets.
* Using the default parameters, fit the [XGBClassifier](https://xgboost.readthedocs.io/en/latest/python/python_api.html#module-xgboost.sklearn) estimator to the training set and use the model to predict values in the test set.
* Evaluate the performance of the classifier using the accuracy score, f-1 score, and confusion matrix from [sklearn.metrics](https://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics).

**Task 4: Univariate Feature Selection**

* In univariate feature selection, we will use the [SelectKBest()](https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html) function from scikit-learn. The score returned can be used to select *n* features with the highest values for the test chi-squared statistic from the data.
* Recall that the chi-square test measures the dependence between stochastic variables.
* Using this function weeds out the features that are the most likely to be independent of class and therefore irrelevant for classification.

**Task 5: Recursive Feature Elimination with Cross-Validation**

* In this task, we will not only find the best features but also the optimal number of features needed for the best classification accuracy.

**Task 6: Plot CV Scores vs Number of Features Selected**

* Evaluate the the optimal number of features needed for the highest classification accuracy by plotting the cross validation (CV) scores of the selected features on the y-axis, against the number of selected features on the x-axis.

**Task 7: Feature Extraction using Principal Component Analysis**

* Use principle component analysis (PCA) for feature extraction.
* We will first need to normalize the data for better performance.
* A plot of the cumulative explained variance against the number of components gives the percentage of variance explained by each of the selected components. This curve quantifies how much of the total variance is contained within the first *N* components.