**Detect Overfitting and Underfitting with Learning Curves**

For this quiz, we'll be using three models to train the circular dataset below.

* A Decision Tree model,
* a Logistic Regression model, and
* a Support Vector Machine model.

Chart, scatter chart

Description automatically generated

One of the models overfits, one underfits, and the other one is just right. First, we'll write some code to draw the learning curves for each model, and finally we'll look at the learning curves to decide which model is which.

First, let's remember that the way the curves look for the three models, is as follows:

Chart, line chart

Description automatically generated

For the first part of the quiz, all you need is to uncomment one of the classifiers, and hit 'Test Run' to see the graph of the Learning Curve. But if you like coding, here are some details. We'll be using the function called learning\_curve:

train\_sizes, train\_scores, test\_scores = learning\_curve(

estimator, X, y, cv=None, n\_jobs=1, train\_sizes=np.linspace(.1, 1.0, num\_trainings))

No need to worry about all the parameters of this function (you can read some more in [here](http://scikit-learn.org/stable/auto_examples/model_selection/plot_learning_curve.html), but here we'll explain the main ones:

* estimator, is the actual classifier we're using for the data, e.g., LogisticRegression() or GradientBoostingClassifier().
* X and y is our data, split into features and labels.
* train\_sizes are the sizes of the chunks of data used to draw each point in the curve.
* train\_scores are the training scores for the algorithm trained on each chunk of data.
* test\_scores are the testing scores for the algorithm trained on each chunk of data.

Two very important observations:

* The training and testing scores come in as a list of 3 values, and this is because the function uses 3-Fold Cross-Validation.
* **Very important:** As you can see, we defined our curves with Training and Testing **Error**, and this function defines them with Training and Testing **Score**. These are opposite, so the higher the error, the lower the score. Thus, when you see the curve, you need to flip it upside down in your mind, in order to compare it with the curves above.

**Part 1: Drawing the learning curves**

In here, we'll be comparing three models:

* A **Logistic Regression** model.
* A **Decision Tree** model.
* A **Support Vector Machine** model with an rbf kernel, and a gamma parameter of 1000 (this is another type of model, don't worry about how it works for now).

Uncomment the code for each one, and examine the learning curve that gets drawn. If you're curious about the code used to draw the learning curves, it's on the **utils.py** tab.