Failure Prediction Application Guide

Introduction

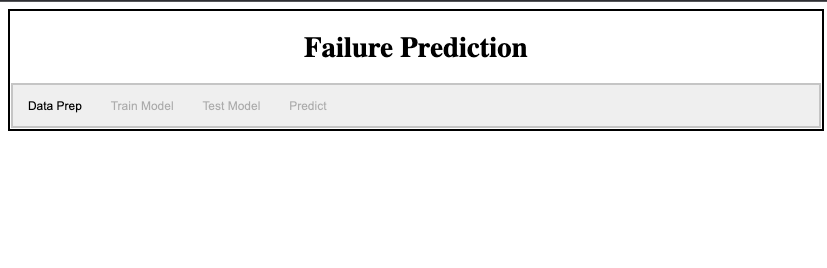
This application serves as a wizard that allows the user to:

* Prepare data for training, testing and prediction
* Train the model
* Test the model
* Make predictions on data that the user chooses

The goal of this application is to ultimately use a trained model that has learned what failures look like to specify there is a possibility of an imminent failure.

Opening Screen

When you first open the application, you will see **tabs**, some of which are initially disabled:



This user interface will guide you through all the processes from Data Preparation through Predictions and will make sure that you perform all processes in the correct order, just like a wizard.

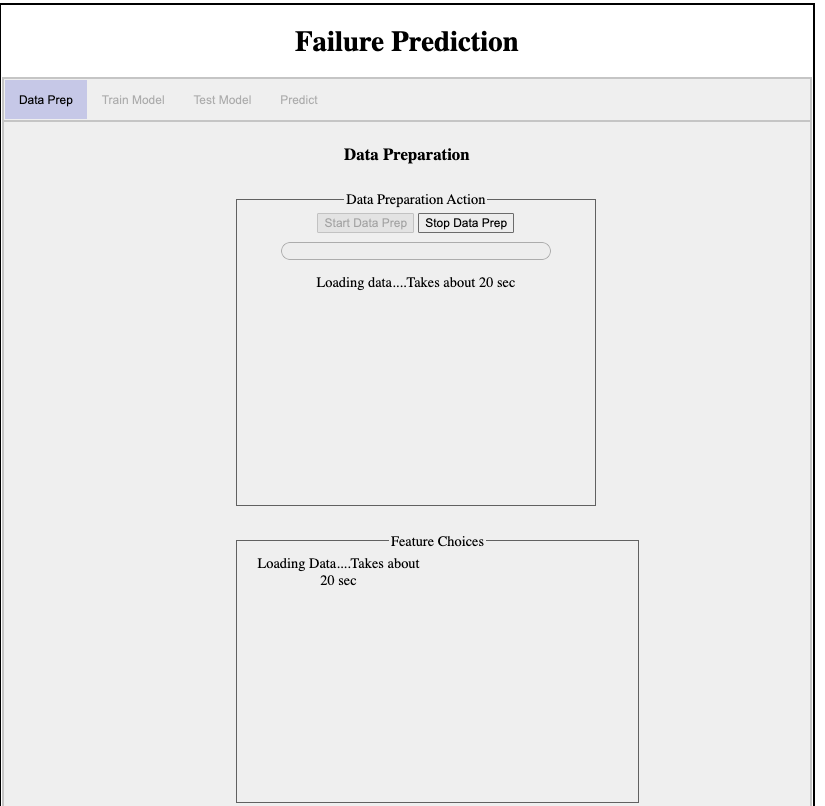
The gray bar with the labels Data Prep, Train Model, Test Model, Predict serve as navigation tabs. When you click on a tab, the relevant screen will appear. Think of the tabs as menu selections that change the interface to suit the job at hand.

Data Preparation

When the application first loads in your browser, a large amount of data is loaded.

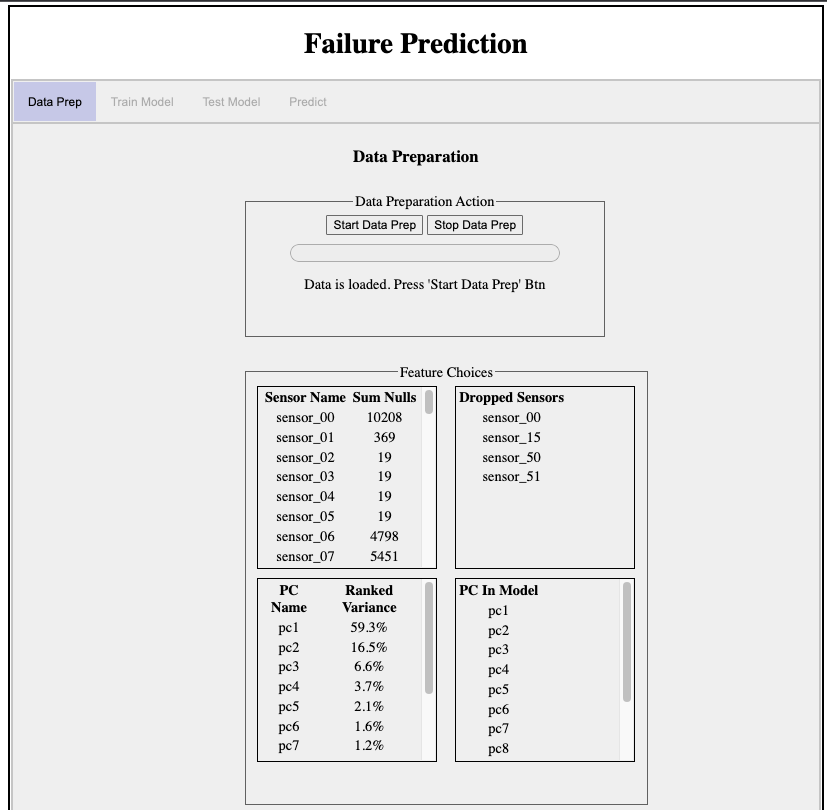
**Click on the Data Prep tab**.

If you click on the Data Prep tab before all the data has been loaded, you will see:



The message: Loading data…. means that you cannot proceed past this page until the data are loaded. Notice that the Start Data Prep button is disabled, and will stay disabled until the data are loaded.

Once loaded, the Start Data Prep button will be enabled.



The information that you see in the box labeled Feature Choices refers to what you discovered in the start of the workshop. That is, you did some data discovery. The box in the upper left shows how many nulls were found in each sensor column. That is why, for example, the box in the upper right shows a list of the sensors that will be dropped, including sensor\_00.

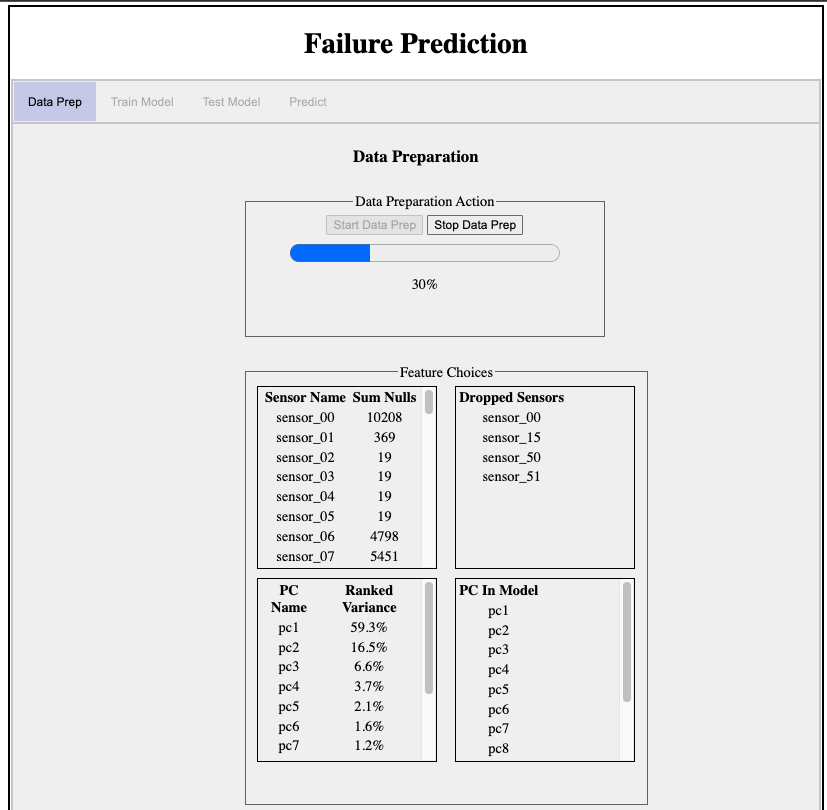
The box in the lower left shows a list of results from the Principal Component Analysis (PCA) that you found in the data discovery. In the original data there were about 52 sensors. We would like to shorten this list to make the model training take less time. Instead of going through a long process of deciding which sensors to keep, we use the PCA that finds which linear combinations of all the sensors gives the most variance. The linear combinations are denoted by pc1, pc2, pc3, …, ( called Principal Components) where pc1 is the first ranked. We then take however many pc’s we need so that the sum of their ranked variances is at least 95%. In our case it takes 12 pc’s to reach that goal. We then use those 12 pc’s to represent a transformation of our original 52 sensors when we train the model.

Now that we have all the data loaded and the PCA transformation is complete, we are ready to proceed with preparing the data.

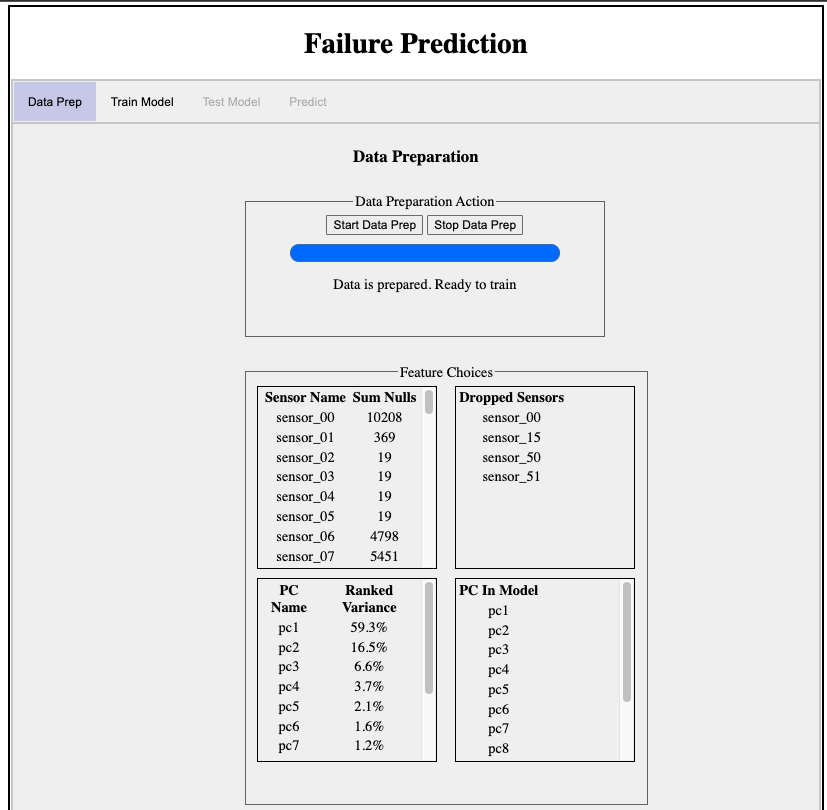
**Click on the Start Data Prep button**.

Each row of data that we will use has a timestamp together with the 12 pc’s that we will now call model features. Data that is ordered by time is called a Time Series.

In the background, the application is reshaping the time series data so that it will be put into a form that the model understands. While this processing is taking place you will see:

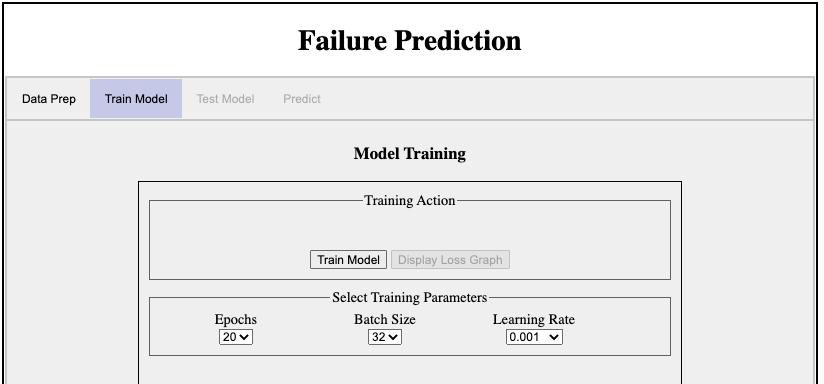


The progress bar will start slowly and at about 50% will speed up. When the data preparation is complete, a message will appear and the next tab, Train Model tab will be enabled:



Training the Model

**Click on the Train Model tab.**

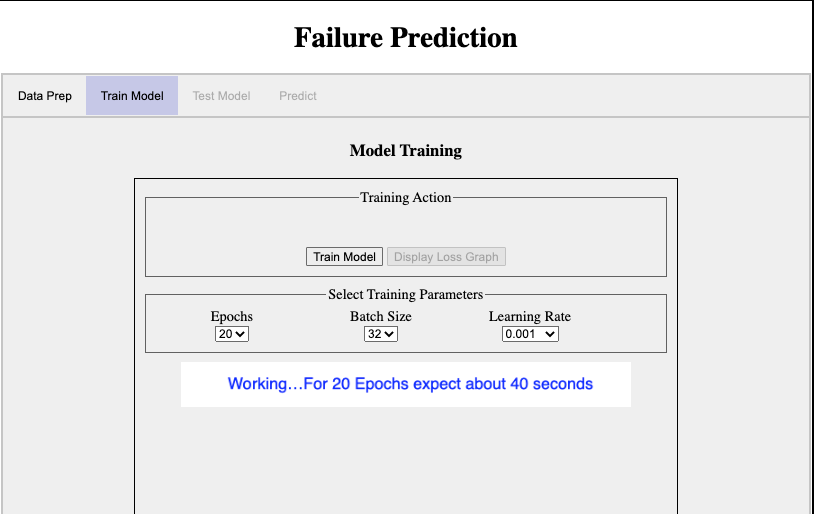


The training process of a model is sometimes called the learning process. This learning process is iterative. A model is a numerical approximation of an entity, and the training data are used to calculate coefficients of the approximation. The training data contain values for the features (input) together with values for the target (output). So during one pass ( called an **epoch**) at the data, small samples of the training data ( called a **batch**) are used to update the model’s coefficients. During training, a parameter called the learning rate is used to determine the “pace” of the iterative steps used to update the coefficients. If the learning rate is too large, then the model may step over a critical part of the updating process. If too small, the updates may cause the model to converge too quickly and possibly miss the optimal approximation. In this training session, the learning rate starts as the value in the pull down, but after 10 epochs its value is divided by a factor of 10.

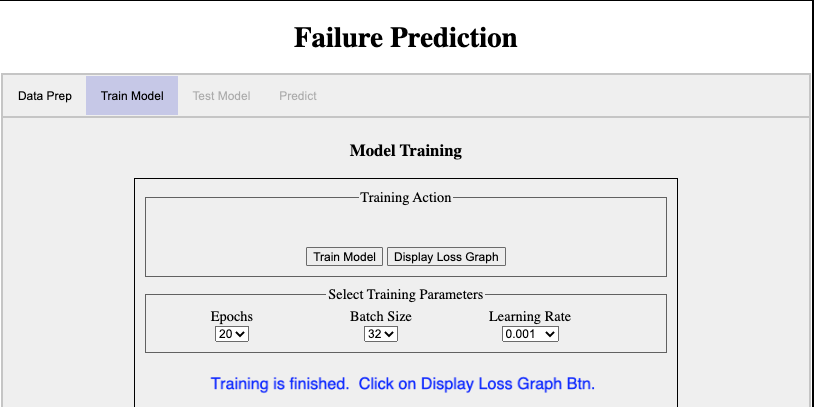
The default values in the pull downs are better left alone for this workshop.

**Click on the Train Model button to start the training.**

During training, there is no progress feedback. This is because the fit() function only gives progress in the form of text displayed in a terminal console. Since a web application has no console, there is no progress feedback available.

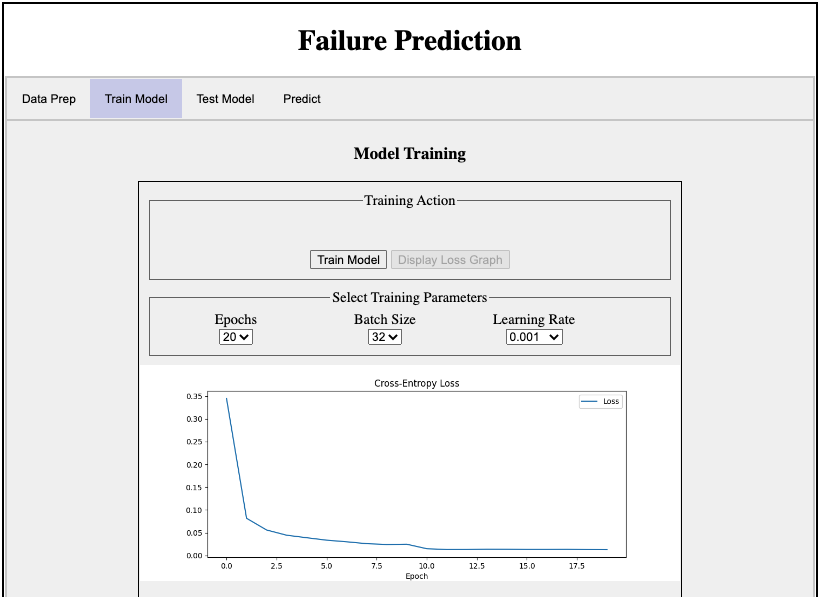


When the training is finished you will see:



Notice that the Display Loss Graph button is now enabled.

**Click on the Display Loss Graph button** and we will see a graph showing the Loss Graph:



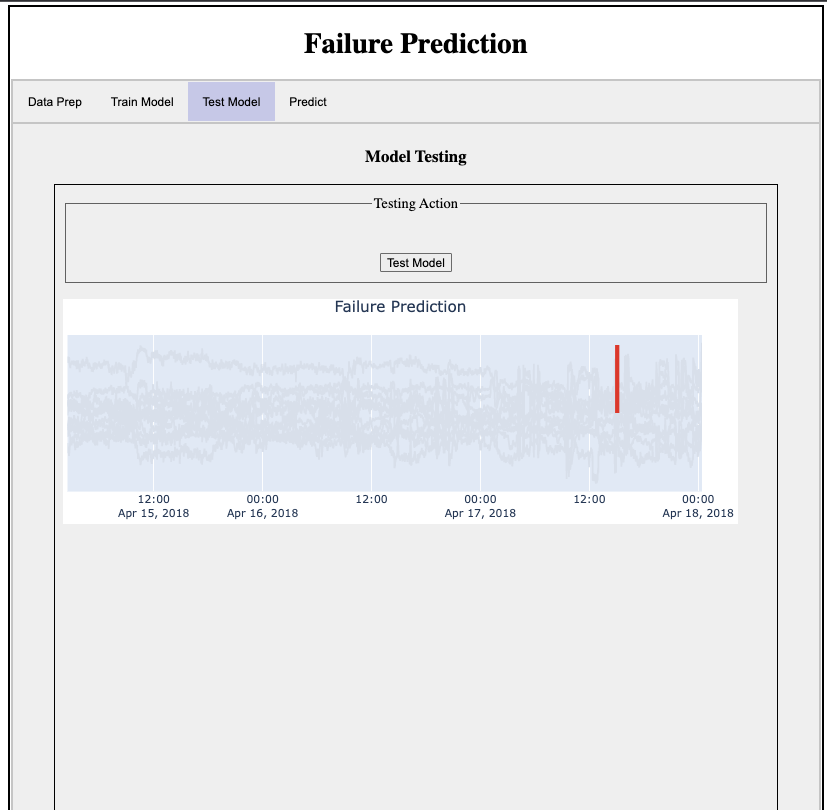
Besides the graph you will notice that the two tabs labeled Test Model and Predict have been enabled. We will run the Test first, although at this point we could just as well run a prediction.

Testing the Model

**Click on the Test Model tab.**

The Test Model tab is sparse because there are no parameters to define for training.

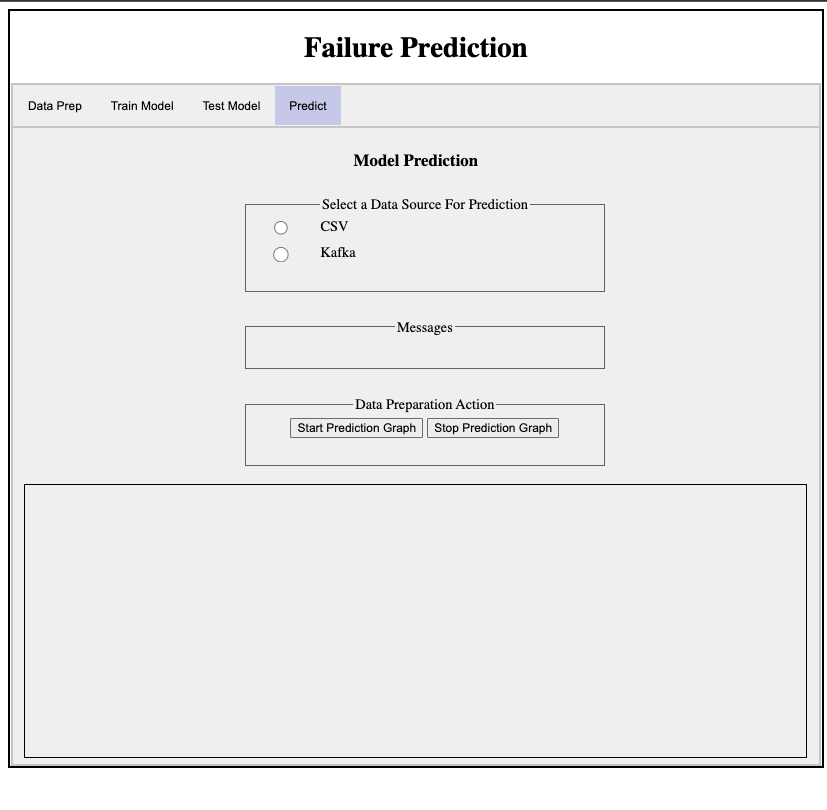
Just **click on the Test Model button.** The result is something like:



Do not be surprised if your neighbor gets a slightly different result. This is because of all the moving parts of the process, that is, because of the stochastic nature of training the model. Remember that a model is an approximation and the training involves uncertainties and randomness in order to make the approximation.

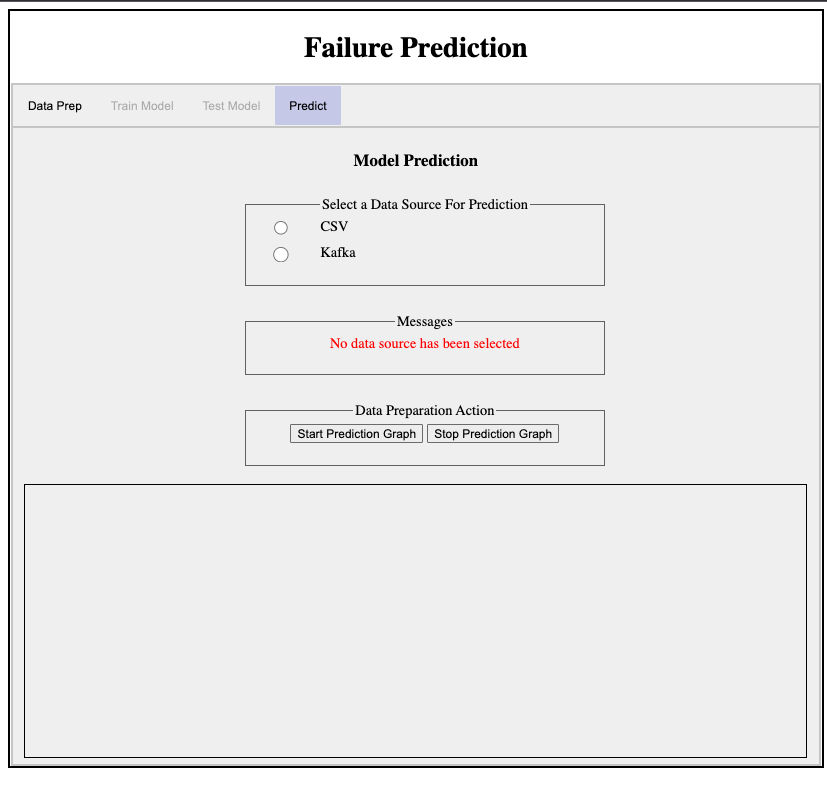
Prediction With the Model

**Click on the Prediction tab**:



Before we can make a prediction, we must first select a data source for the prediction data. There are two options:

1. Use a CSV file which will be streamed one point at a time, simulating real time generation. The data in the CSV file is taken from the original Kaggle data source as test data.
2. Use a data stream of *synthetic* data with the help of Apache Kafka that also simulates the production of real time data. NOTE: **You cannot run a prediction if no data source has been selected.** If you attempt to click on the Start Prediction Graph before selecting a data source you will get a red message:



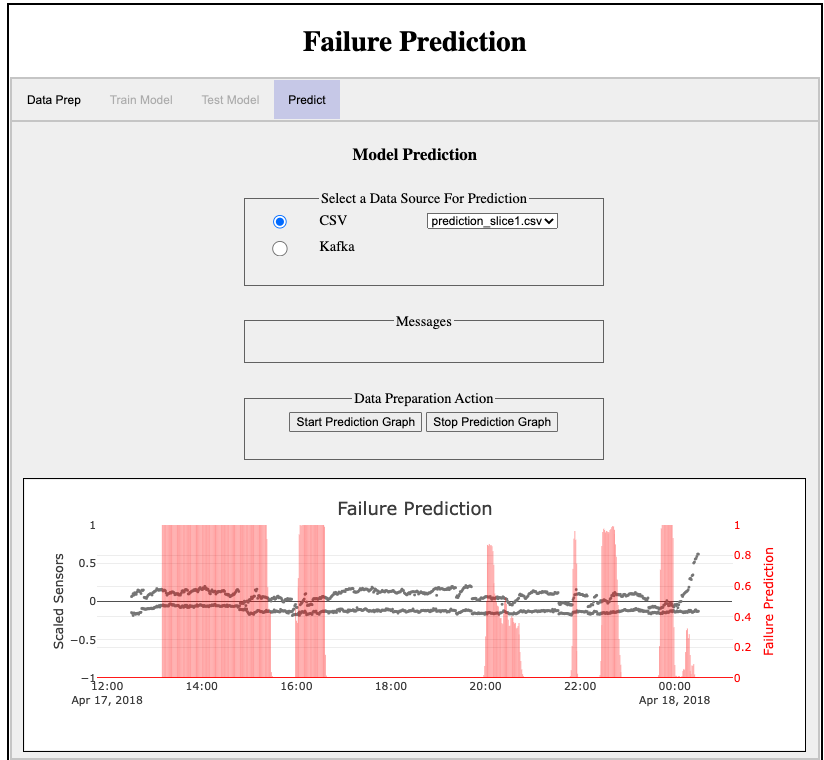
*If you choose the CSV radio button*, a select box will list CSV file names available. Just **select a CSV filename**.

*If you chose the Kafka ratio button*, **enter the Group Id** that you have been given. This Group Id allows your browser session to pull out messages that you were identified with.

*The purpose of the button labeled Stop Prediction Graph is to allow you to stop a prediction process before it is finished. You can use this button if you wish to choose another data source and run the prediction again before the last prediction process has finished.*

**Click on the Start Prediction Graph button**.

There will be a pause while the data stream is prepared. Then you will see points from the prediction data source as well as a red prediction alarm if the model predicts failure is imminent. Note that the prediction data is only available 12 hours before failure.



In this case actual failure was at 2018-04-18 00:30. The graph in your browser is interactive, so you can move your cursor to display the times.