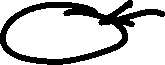
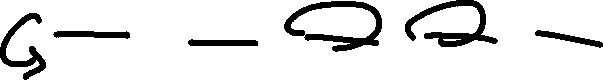
**ighProblem Statement**

To build a classification methodology to predict the quality of wafer sensors based on the given training data.

**Architecture**





**Data Description**

The client will send data in multiple sets of files in batches at a given location. Data will contain Wafer names and 590 columns of different sensor values for each wafer. The last column will have the "Good/Bad" value for each wafer.

"Good/Bad" column will have two unique values +1 and -1.

"+1" represents Bad wafer.

"-1" represents Good Wafer.

Apart from training files, we also require a "schema" file from the client, which contains all the relevant information about the training files such as:

Name of the files, Length of Date value in FileName, Length of Time value in FileName, Number of Columns, Name of the Columns, and their datatype.

**Data Validation**

In this step, we perform different sets of validation on the given set of training files.

1. Name Validation- We validate the name of the files based on the given name in the schema file. We have created a regex pattern as per the name given in the schema file to use for validation. After validating the pattern in the name, we check for the length of date in the file name as well as the length of time in the file name. If all the values are as per requirement, we move such files to "Good\_Data\_Folder" else we move such files to "Bad\_Data\_Folder."
2. Number of Columns - We validate the number of columns present in the files, and if it doesn't match with the value given in the schema file, then the file is moved to "Bad\_Data\_Folder."
3. Name of Columns - The name of the columns is validated and should be the same as given in the schema file. If not, then the file is moved to "Bad\_Data\_Folder".
4. The datatype of columns - The datatype of columns is given in the schema file. This is validated when we insert the files into Database. If the datatype is wrong, then the file is moved to "Bad\_Data\_Folder".
5. Null values in columns - If any of the columns in a file have all the values as NULL or missing, we discard such a file and move it to "Bad\_Data\_Folder".

**Data Insertion in Database**

1) Database Creation and connection - Create a database with the given name passed. If the database is already created, open the connection to the database.

2) Table creation in the database - Table with name - "Good\_Data", is created in the database for inserting the files in the "Good\_Data\_Folder" based on given column names and datatype in the schema file. If the table is already present, then the new table is not created and new files are inserted in the already present table as we want training to be done on new as well as old training files.

3) Insertion of files in the table - All the files in the "Good\_Data\_Folder" are inserted in the above-created table. If any file has invalid data type in any of the columns, the file is not loaded in the table and is moved to "Bad\_Data\_Folder".

**Model Training**

1) Data Export from Db - The data in a stored database is exported as a CSV file to be used for model training.

2) Data Preprocessing

a) Check for null values in the columns. If present, impute the null values using the KNN imputer.

b) Check if any column has zero standard deviation, remove such columns as they don't give any information during model training.

3) Clustering - KMeans algorithm is used to create clusters in the preprocessed data. The optimum number of clusters is selected by plotting the elbow plot, and for the dynamic selection of the number of clusters, we are using "KneeLocator" function. The idea behind clustering is to implement different algorithms

To train data in different clusters. The Kmeans model is trained over preprocessed data and the model is saved for further use in prediction.

4) Model Selection - After clusters are created, we find the best model for each cluster. We are using two algorithms, "Random Forest" and "XGBoost". For each cluster, both the algorithms are passed with the best parameters derived from GridSearch. We calculate the AUC scores for both models and select the model with the best score. Similarly, the model is selected for each cluster. All the models for every cluster are saved for use in prediction.

**Prediction Data Description**

Client will send the data in multiple set of files in batches at a given location. Data will contain Wafer names and 590 columns of different sensor values for each wafer.

Apart from prediction files, we also require a "schema" file from client which contains all the relevant information about the training files such as:

Name of the files, Length of Date value in FileName, Length of Time value in FileName, Number of Columns, Name of the Columns and their datatype.

**Data Validation**

In this step, we perform different sets of validation on the given set of training files.

1) Name Validation- We validate the name of the files on the basis of given Name in the schema file. We have created a regex pattern as per the name given in schema file, to use for validation. After validating the pattern in the name, we check for length of date in the file name as well as length of time in the file name. If all the values are as per requirement, we move such files to "Good\_Data\_Folder" else we move such files to "Bad\_Data\_Folder".

2) Number of Columns - We validate the number of columns present in the files, if it doesn't match with the value given in the schema file then the file is moved to "Bad\_Data\_Folder".

3) Name of Columns - The name of the columns is validated and should be same as given in the schema file. If not, then the file is moved to "Bad\_Data\_Folder".

4) Datatype of columns - The datatype of columns is given in the schema file. This is validated when we insert the files into Database. If dataype is wrong then the file is moved to "Bad\_Data\_Folder".

5) Null values in columns - If any of the columns in a file has all the values as NULL or missing, we discard such file and move it to "Bad\_Data\_Folder".

**Data Insertion in Database**

1) Database Creation and connection - Create database with the given name passed. If the database is already created, open the connection to the database.

2) Table creation in the database - Table with name - "Good\_Data", is created in the database for inserting the files in the "Good\_Data\_Folder" on the basis of given column names and datatype in the schema file. If table is already present then new table is not created, and new files are inserted the already present table as we want training to be done on new as well old training files.

3) Insertion of files in the table - All the files in the "Good\_Data\_Folder" are inserted in the above-created table. If any file has invalid data type in any of the columns, the file is not loaded in the table and is moved to "Bad\_Data\_Folder".

**Prediction**

1) Data Export from Db - The data in the stored database is exported as a CSV file to be used for prediction.

2) Data Preprocessing

a) Check for null values in the columns. If present, impute the null values using the KNN imputer.

b) Check if any column has zero standard deviation, remove such columns as we did in training.

3) Clustering - KMeans model created during training is loaded, and clusters for the preprocessed prediction data is predicted.

4) Prediction - Based on the cluster number, the respective model is loaded and is used to predict the data for that cluster.

5) Once the prediction is made for all the clusters, the predictions along with the Wafer names are saved in a CSV file at a given location and the location is returned to the client.

**Deployment**

We will be deploying the model to the Pivotal Cloud Foundry platform.

This is a workflow diagram for the prediction of using the trained model.

**Now let’s see the Wafer fault Detection project folder structure.**



**requirements.txt** file consists of all the packages that you need to deploy the app in the cloud.



**main.py** is the entry point of our application, where the flask server starts. Here we will be decoding a base64 to an image, and then we will be making predictions.



This is the **obj.py** file where the predictions take place based on the image we are giving input to the model.



**manifest.yml**:- This file contains the instance configuration, app name, and build pack language.

****

**Procfile** :- It contains the entry point of the app.

****

**runtime.txt**:- It contains the Python version number.

****

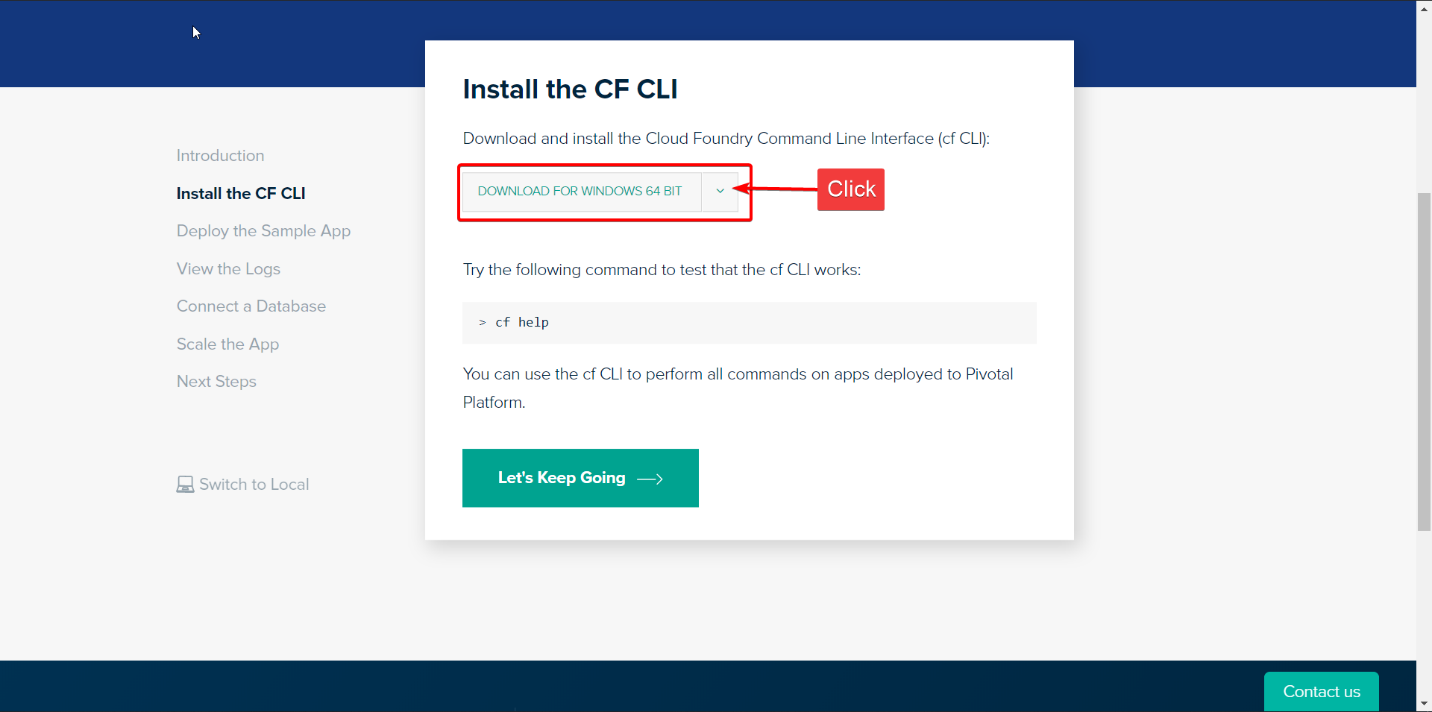
Visit the official website [**https://pivotal.io/platform**](https://pivotal.io/platform)**.**

****

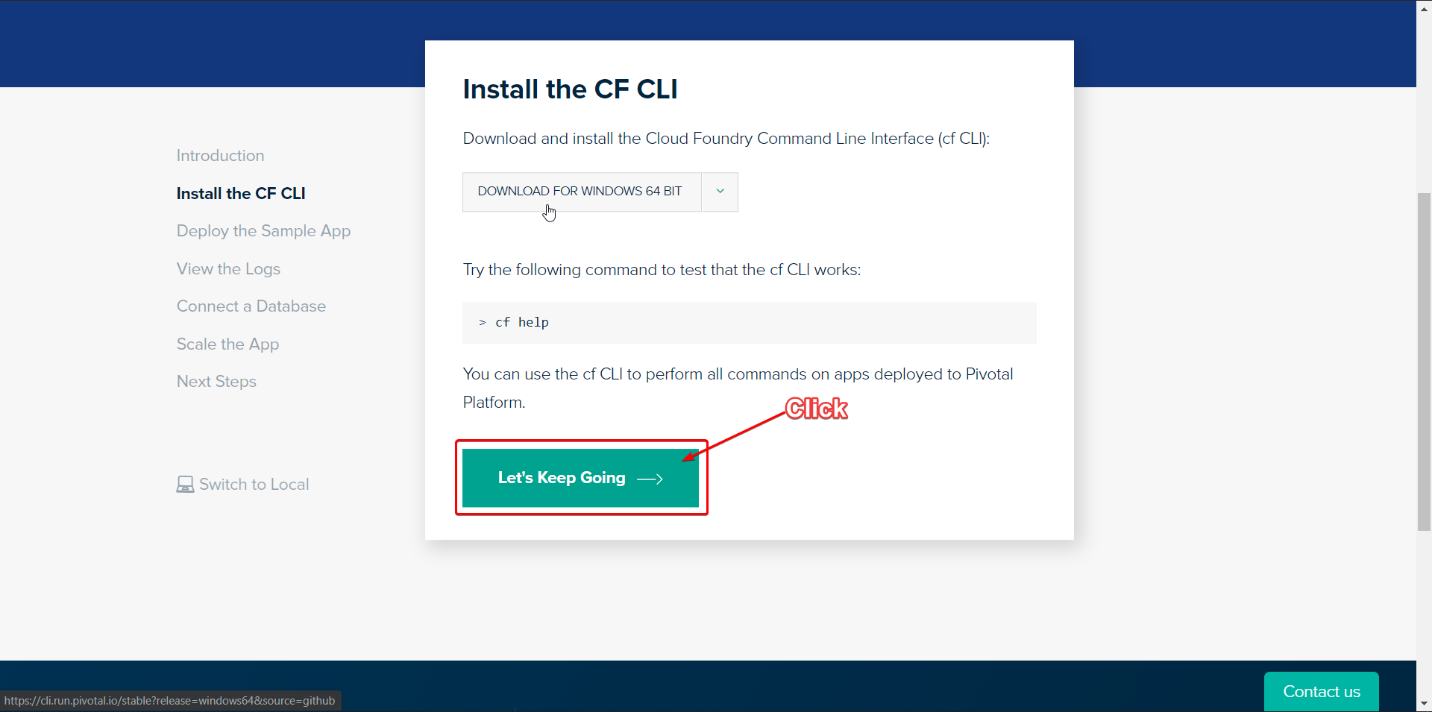
Scroll Down to see the **Start Trial Button**

****

Click on the start trial button and the next interface will open. Then I will click on **I’m ready to continue**

****

Click on Download for **Windows 64 bit, and** then zip file will be downloaded. Keep it for future uses.

****

Now click on **Let’s Keep Going**

****

Click on **Create Your Account**

****

Fill Up your Details For registration

Do the email verification

Then login in again

****

After logging you will see this screen below and start your free trial.

Write any Company or which one you prefer

****

Enter your **Mobile Number** for Verification

****

Click on **Pivotal Web Services**

****

Give any **Org** name

****

Now you are inside your Org, and by default, development space is created in your org. You can push your apps here.

The cloud signup process is done, and the setup is ready for us to push the app.

Previously you have downloaded the **CLI.zip** file. Unzip the file and install the .exe file with admin rights.

After a successful installation, you can verify by opening your CMD and type **cf**.

Then you will get a screen which is shown below

****

If you see this screen in your CMD, the installation is successful.

Now type the command to login via cf-cli

cf login -a <https://api.run.pivotal.io>

Next, enter your email and password. Now you are ready to push your app.

Now let’s go to the app which we have built.



Navigate to the project folder after downloading from the given below link:-

Then write cf push in the terminal.



After the app is successfully deployed in the cloud, you will see the screen below with the route.

****

Finally, the app is pushed in the cloud.

Lets Open Postman and see the result.

****