**Water Pump Failure Prediction**

*A project report submitted to ICT Academy of Kerala*

*in partial fulfillment of the requirements*

*for the certification of*

**CERTIFIED SPECIALIST**

**IN**

**DATA SCIENCE & ANALYTICS**

submitted by

**Team 6**

**Members**

**Helin M Jibu, Archana P S, Christo Jestine, Anjali J, Mariya Johnson**



**ICT ACADEMY OF KERALA**

**THIRUVANANTHAPURAM, KERALA, INDIA**

**Nov 2022**

**List of Figures**

|  | **Figures** | **Page No** |
| --- | --- | --- |
| 3.1 | Histogram showing the sensor value readings | 8 |
| 3.2 | Heatmap of the sensor values | 9 |
| 3.3 | Countplot of Machine \_status(Target column) | 10 |
| 3.4 | Correlation plot of the sensors | 11 |
| 4.1 | Histogram of the column**s** | 12 |
| 4.2 | Heatmap before and after handling null values | 13 |
| 4.3 | Boxplot showing the Outliers | 14 |
| 4.4 | Fig showing boxplot line meanings | 14 |
| 4.5 | Sensor\_01 before and after outlier handling | 15 |
| 4.6 | Sensor\_01 and Sensor\_04 before Standard Scaling | 16 |
| 4.7 | Sensor\_01 and Sensor\_04 after Standard Scaling | 17 |
| 5.1 | Train and Test split of machine\_status column | 19 |
| 5.2 | Confusion matrix of Logistic Regression model | 20 |
| 5.3 | Classification Report of Logistic Regression Model | 21 |
| 5.4 | Confusion matrix of Logistic Decision Tree model | 21 |
| 5.5 | Classification Report of Decision Tree Model | 22 |
| 5.6 | Confusion matrix of Random Forest model | 22 |
| 5.7 | Classification Report of Random Forest Model | 23 |
| 5.8 | Confusion matrix of SVM (linear) model | 23 |
| 5.9 | Classification Report of SVM (linear) Model | 24 |
| 5.10 | Confusion matrix of SVM(rbf) model | 24 |
| 5.11 | Classification Report of SVM(rbf) Model | 24 |
| 5.12 | K value | 25 |
| 5.13 | Classification Report of knn Model | 25 |
| 6.1 | Model.py | 26 |
| 6.2 | HTML Page | 27 |
| 6.3 | Website after doing CSS | 28 |
| 6.4 | Pythonanywhere | 30 |
| 6.5 | RECOVERING Condition | 31 |
| 6.6 | NORMAL Condition | 31 |
|  |  |  |

**Table of Contents**

|  | **Content** | **Page No** |
| --- | --- | --- |
|  | Abstract | 5 |
| 1 | Problem definition | 6 |
| 2 | Introduction | 8 |
| 3 | Exploratory data analysis | 9 |
| 4 | Preprocessing | 12 |
| 5 | Model building | 18 |
| 6 | Web Hosting | 26 |
| 7 | Result | 31 |
| 8 | Conclusion | 32 |
| 9 | References | 33 |

**Abstract**

Sensor data plays a vital role in predicting the health of machines. This may be the key reason why there are a number of sensors attached to each and every electronic instrument in order to monitor the working condition and predict the failure in advance. The best example to showcase the importance of sensor data is the Electronic Control unit scanning which is now popular in the automobile industry to monitor health and predict failures in advance.

As a part of our project we have taken into consideration the sensor data which predict the health condition of the Pump. This dataset contains 52 sensor readings which were used to monitor the conditions and 22000 instances which will help us to predict the health condition based on this record. The three basic target criteria of our data is Normal, Broken and Recovering. Our project mainly lies on processing the data and creating a machine learning model which will be able to predict the status of the machine by inputting the sensor values. We are also planning to have a front end model. This prediction can in turn help us to determine the machine status and prevent the failure by scheduling a proper maintenance plan.

Keywords :- Sensors, Failure, pump

**1. Problem Definition**

**1.1 Overview**

The pump dataset is made after doing the preprocessing steps to create an efficient model with much greater accuracy and F1 score. Later this model will be used as the back end of the web app which we are planning to develop to predict the working condition under the different sensor values. This prediction will help us to understand the health of the machine and do the necessary steps to maintain the health. The role played by the sensor data to save the instruments from un predictable failures and increase the overall income of the firm is very high. Therefore the prediction mode by the model can help the user to avoid unpredictable losses.

**1.2 Problem Statement**

In the recent era of technology , the dominant role of sensors provides a solution to a wide variety of problems .Sensor data is adding a great value by giving us large insights about the real time machine conditions .

So, in our project we are going to make use of this sensor data in order to predict the working condition and help the user to find out the condition of the machine with the help of the front end model. In this project we are making use of a pump dataset inorder to train the model and to make predictions on the working condition of the pump with help of this model. Predicting the working condition of a pump with the help of a ML model build based on the sensor dataset available.

In our project we have taken into consideration the sensor data which predict the health condition of the Pump. This dataset contains 52 sensor readings which were used to monitor the conditions and 22000 instances which will help us to predict the health condition based on this record and also create a front end model.

**2. Introduction**

In every system whether in domestic or industry, we need to monitor the pumps and also should maintain them properly. We use sensors to record temperature, pressure, vibration, load capacity, volume, flow density etc. These are only initial investments to set up the data collection process and feed the collected data into the ML model to identify pump failure.

Here in our project we are using the pump- sensor data to predict the pump failure. The dataset should undergo many steps for making the best front end model. EDA( Exploratory Data analysis) , Data Preprocessing, Model Building, Creating Front End are the basic steps to follow.

EDA ( Exploratory Data Analysis) is an approach to analyze the data using visual techniques. It is used to discover trends, patterns or to check assumptions with the help of statistical summary and grapical representations.

Data preprocessing is the step where preprocessing of data in order to remove any faulty data, null values or any thing which may negatively impact our ML model and making it suitable for building machine learning model

Model building is a step where we are trying different regression models to predict the output and selecting a model which gives more accurate prediction values.

Creating a front end is the last step of the project.In the front end we will be hosting a website that displays the condition of the pump by intaking sensor values from the user .

The project topic that we have taken is very much relevant these days. It makes the job of the engineers a little easier and will protect from huge capital loss.

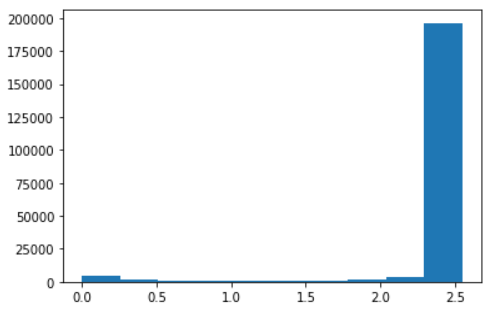
**3.EDA(Exploratory Data Analysis)**

EDA ( Exploratory Data Analysis) is an approach to analyze the data using visual techniques. It is used to discover trends, patterns or to check assumptions with the help of statistical summary and graphical representations.

We properly understood the data and end up to the following insights :

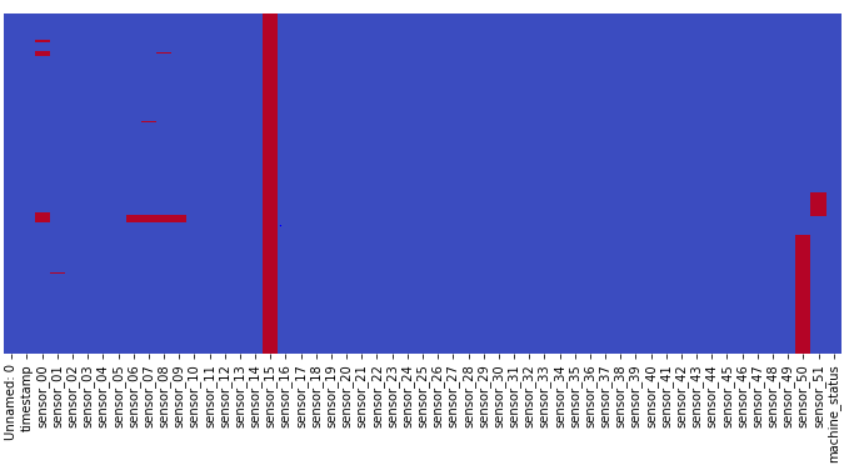
**Basic Understanding of dataset:**

* We displayed the dataset in IDE and understood that the dataset contains 220320 rows and 55 columns .Among the 55 columns we have 52 sensor columns which we are going to use for the predictions [data.shape].
* All the sensor data columns are in float data type and other 3 columns (timestamp,Unnamed;0,machine status) are object data type[data.info()].
* The range of values of each of the sensors are different [data.describe()]



**Fig.3.1. Histogram showing the sensor value readings**

* There is no duplicate values in the dataset [data.duplicated().sum()]
* There are many null values present in the dataset especially in sensor15 no values are present. [ data.isnull().sum()]
* Condition of the sensor\_15 can be easily analyzed with the help of the heat map depicted in the **fig.3.2**

**Fig.3.2. Heatmap of the sensor values** 

* The column Named:0 contains unique values only and is similar to the index values .
* There is no repetition of values in the timestamp.(data[‘timestamp’].nunique() )

**Dropping of columns:**

* Unnamed:0 column ,which is the same as the index value.
* sensor15 column which does not contain any values.
* timestamp column after extracting the features from it.

**Extracting the values from timestamp column:**

For getting more details of the ‘timestamp’ column we extracted the details about

* Year
* Month
* Day
* Time

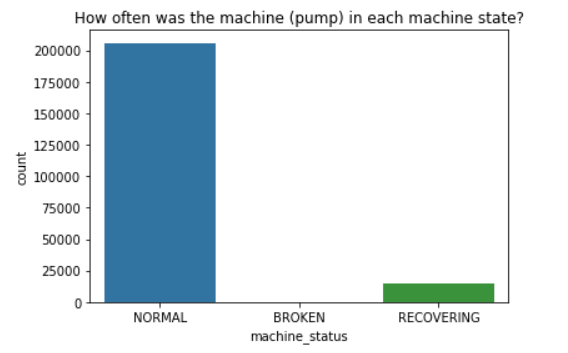
Here is the inference after extracting values from timestamp column

* Data is of a single year that is 2018.
* Data contain the sensor readings from April to August month.
* Dataset contain record of each day of the month
* The reading is taken several times each day.

**Analysis of Target column:**

The target column is the machine status column.We have 3 unique values in the machine status column.

* Normal Condition
* Recovering
* Broken

t

**Fig.3.3.. Countplot of Machine \_status(Target column)**

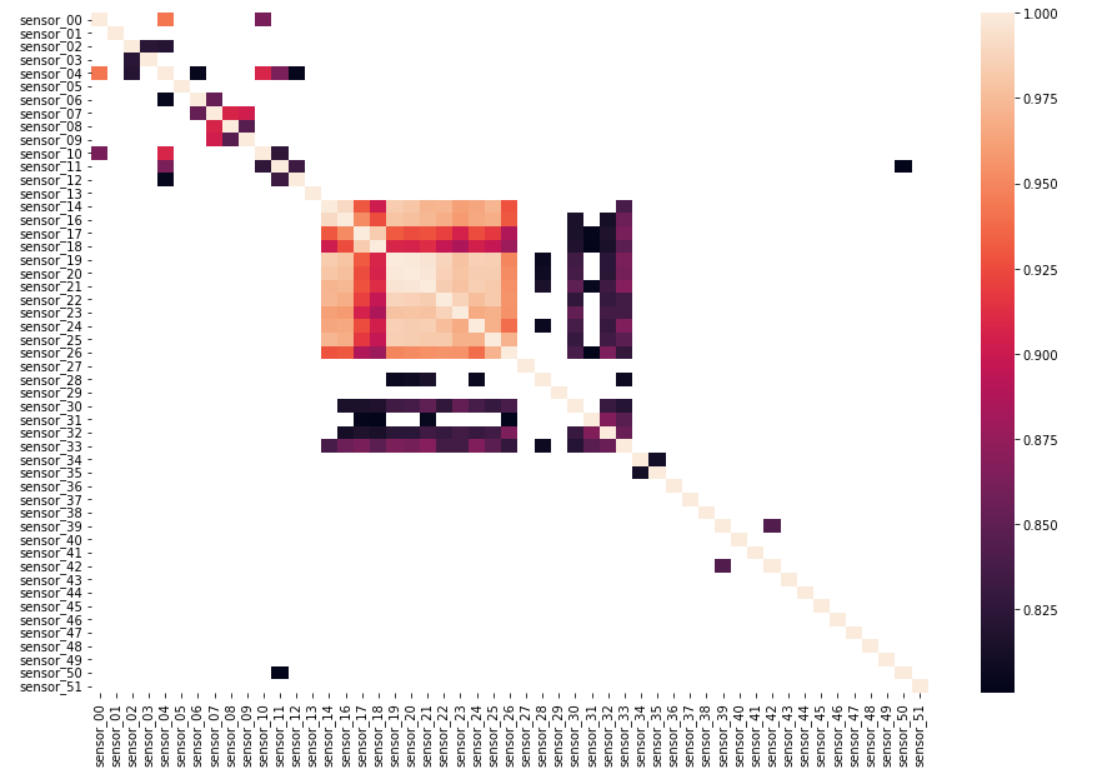
We plotted the countplot for identifying the values related to each .We got that the ‘Broken’ condition happens very rarely (about 7 times).This value is obtained by using the value\_counts function to determine the correct value of the three target states. Since the appearance of a broken stage is very low. It is better to remove the broken condition. The main reason for the removal of the broken condition is that the model cannot learn and make the prediction properly with very very less value count of the broken class.

So we will be dropping the Broken class and dealing with binary class prediction for better performance and better results.

**Correlation Matrix**

Correlation is a statistical relationship between two random variables or bivariate data.There are different correlation coefficients.Most common one is Pearson correlation coefficient .The value of coefficient of correlation ranges from +1 to -1.

Done the correlation matrix of the dataset and found that we have highly correlated sensors (15-25)



**Fig.3.4.Correlation plot of the sensors**

**4.DATA PREPROCESSING**

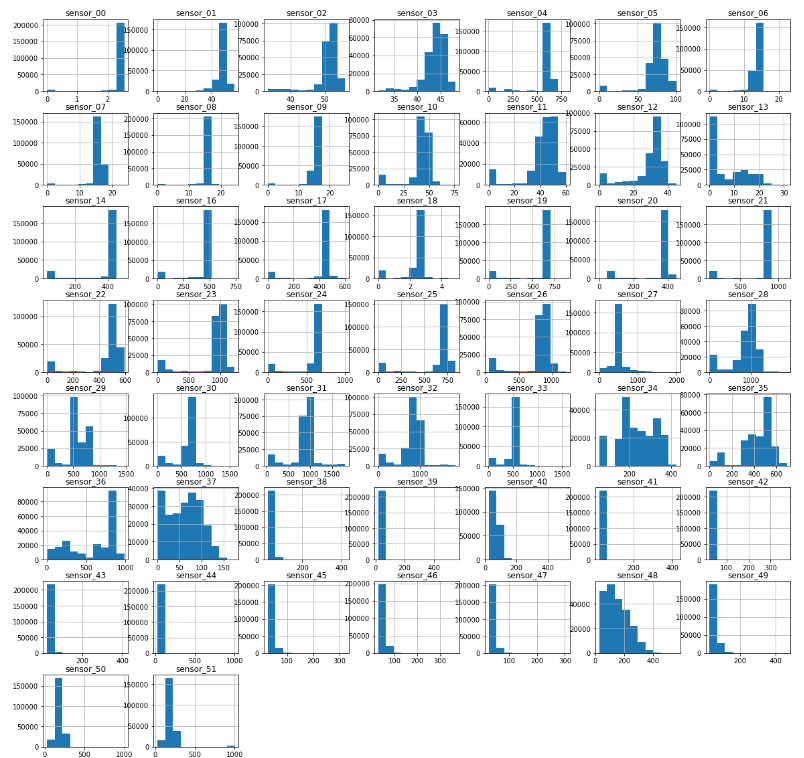
Data preprocessing is the process of transforming raw data into an understandable format. It is also an important step in data mining as we cannot work with raw data. The quality of the data should be checked before applying machine learning or data mining algorithms. Major Tasks in Data Preprocessing are Data cleaning, Data integration, Data reduction and Data transformation.

**4.1. Insights from EDA**

* All sensor columns contain null values.
* Sensor\_15 does not contain any value.
* Unnamed:0 is the same as the index column.
* From the histogram plot of the sensor column, we inferred that the data is skewed.
* The Broken rows should be removed as the count is very low

**4.2.Handling missing values**

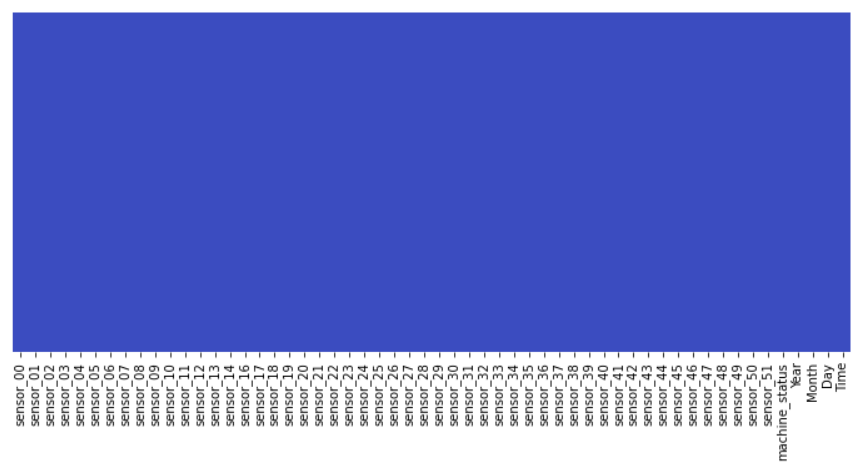
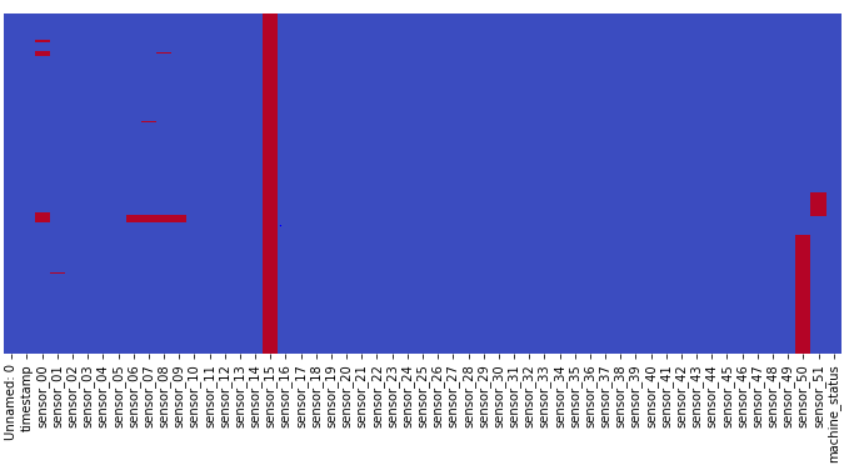
As a part of Data cleaning, we need to remove incorrect data, incomplete data and inaccurate data from the datasets, and it also replaces the missing values.

**Fig 4.1 Histogram of the column**

The sensor\_15 column in the dataset contains no value at all. So the column needs to be dropped.

By plotting the histogram of the sensor data column we found that the data is skewed. .When the data is skewed, it is good to consider using the median value for replacing the missing values. But while consider the time column we can come it to insight of filling the null values with the values in the above column

We filled the missing values using the forward filling method(ffill).This method is taken as the data is a time series data. The heatmap below depicts the change after handling null values.

**Fig 4.2 Heatmap Before and After null value handling**

**4.3 Outlier Handling**

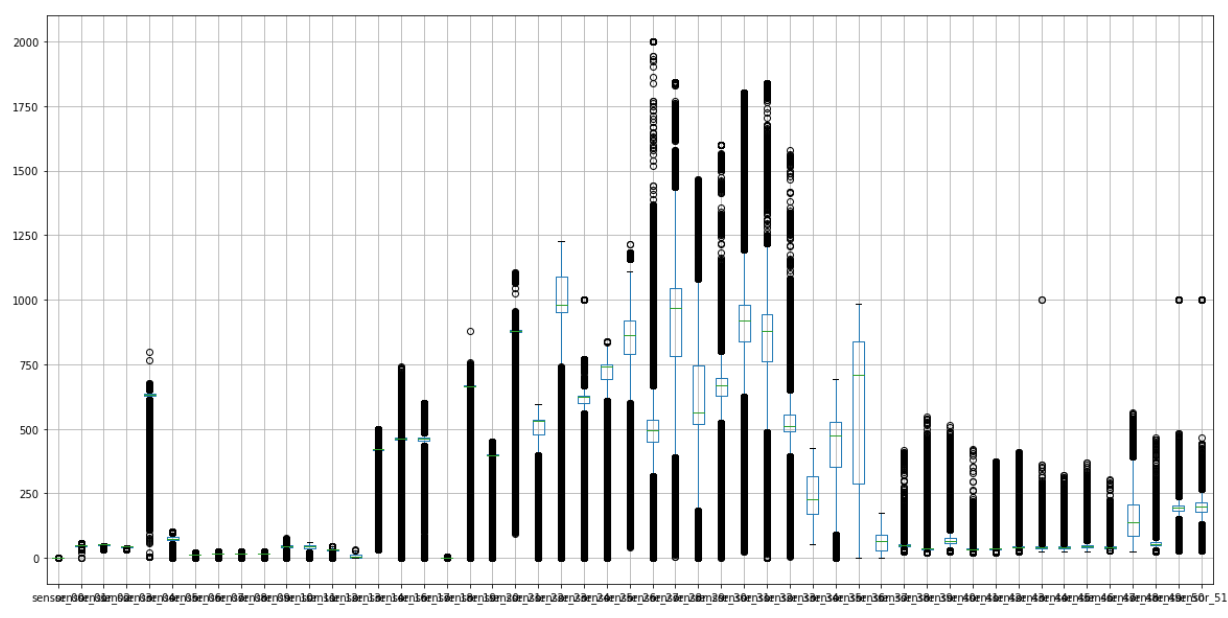
An Outlier is an observation in a given dataset that lies far from the rest of the observations. That means an outlier is vastly larger or smaller than the remaining values in the set.

Detection of outliers can be done using different methods

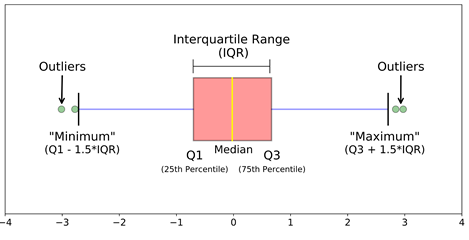
1. Boxplot
2. Z score
3. Interquartile Range

**Detection of outliers using boxplot:**

In descriptive statistics, boxplot is a method for graphically depicting groups of numerical data through their quartiles..Box Plots are a standardized way of displaying the distribution of data based on a five number summary (*“minimum”*, *first quartile (Q1)*, *median*, *third quartile (Q3)*, and *“maximum”*)



**Fig 4.3 Boxplot Showing the Outliers**

**Detection of outliers using interquartile range:**

**Fig 4.4 Fig showing Boxplot line meanings**

Data points that lie 1.5 times of IQR above Q3 and below Q1 are outliers.We did the python coding and detected the outlier values of each sensor in order to handle it in the right way.

**steps we followed in python:**

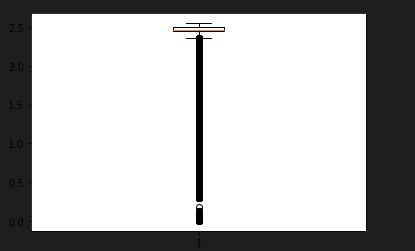
* calculate the 1st and 3rd quartiles(Q1, Q3)
* compute IQR=Q3-Q1
* compute lower bound = (Q1–1.5\*IQR), upper bound = (Q3+1.5\*IQR)
* loop through the values of the dataset and check for those who fall below the lower bound and above the upper bound and mark them as outliers.

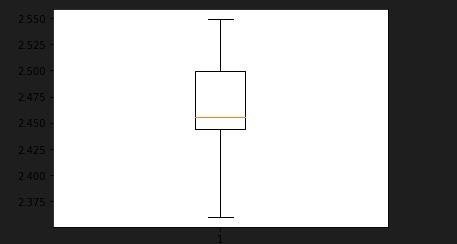
**Handling of outliers :**

There are many methods to handle outliers such as trimming/removing the outliers,imputing it with mean/median,quantile based flooring and capping.In our project we are not trimming or imputing with median since it won’t give a better result in outlier handling. We used quantile based flooring and capping for handling our outliers.

**steps:**

* We have declared a function based on IQR.
* Found the upper and lower and replaced it with the same.
* This Function is called using a for loop through out the Sensor data columns
* The Function replace the value greater than upper with upper and value lesser the lower with lower





**fig 4.5 Sensor\_01 before and after outlier handling**

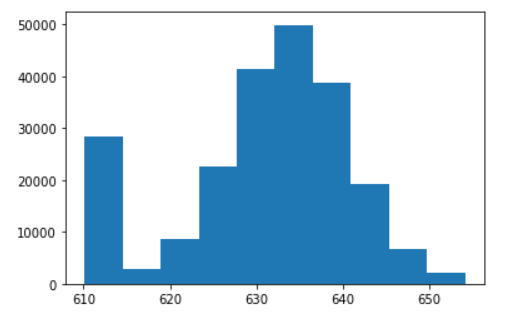
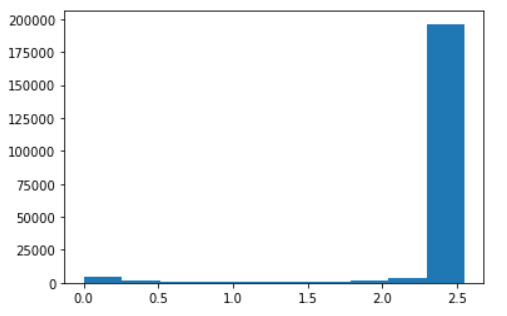
**4.4 Standard Scaling**

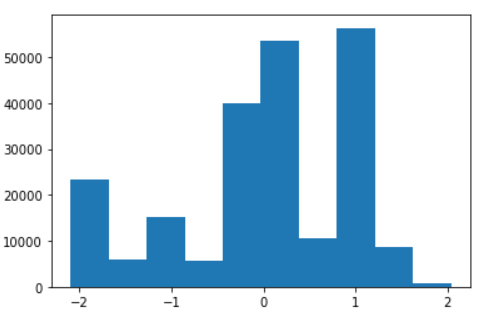
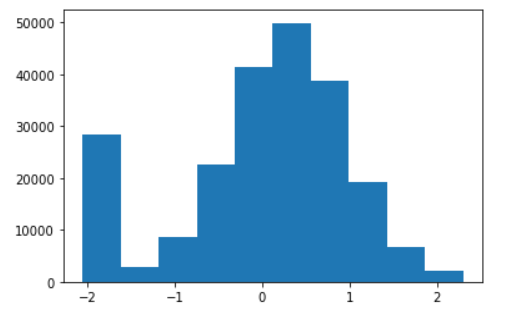
Standardization is an important technique that is mostly performed as a pre-processing step before many machine learning models, to standardize the range of features of an input data set. Since the data we are dealing with contain sensor reading which is in an entirely different range of values. Standardization will result in better results.

The method we have used is Standard Scaler. The standard scaler brings the value between -1 and 1. This is how the data gets standardized and we are able to get more insights from data which in turn also help in increasing the performance of ML models.

To depict the main advantage of scaling visually how the data look after the standardization process we have added before and after the standardization scaling graphs in fig 4.4 and in 4.5. In the figure we can see that before standardization we can see only one value dominating and we could not get any idea about the other values as seen in the fig 4.4(a) after the standardization process we can see or we can identify the unseen peaks which we have missed before. This is the basic representation of the impact of standard scalers in our data.

The Standard scaler is taken from the Sklearn library and the module in which the standard scaler is present is the Preprocessing. We can call Standard Scaler by calling (from sklearn.preprocessing import StandardScaler)

**fig 4.6 Sensor\_01 and Sensor\_04 before Standard Scaling**

**fig 4.7 Sensor\_01 and Sensor\_04 after Standard Scaling**

**4.5 Label Encoding**

Label Encoding refers to converting the labels into a numeric form so as to convert them into the machine-readable form.In our dataset we cannot do the scaling for date and time so we just label encoded them. This process will help the machine to give some meaning to the values of date and time. The label encoding process is done on Year, Month and the day.

**4.6 Dropping of BROKEN classified rows**

We dropped the rows which are classified as BROKEN . When we create a machine learning model it will become biased since the value counts of the BROKEN class is only seven in the entire dataset.So we are dropping the BROKEN classed rows and are going to create a binary classification model. The main reason behind the dropping of the BROKEN is that model cannot understand or learn the broken condition as the value count of BROKEN is very very less

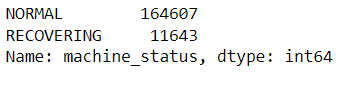
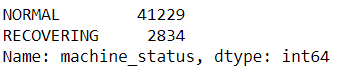
**5.MODEL BUILDING**

In this phase we need to develop data sets for training, testing, and production purposes. These data sets enable us to develop analytical methods and train them, while holding aside some data for testing the model.

**5.1 Train and Test Samples**

We split the dataset into training and testing .Machine Status is our target column .After defining the target column our next step was to split the data into train and testing .The train- test split of data is a procedure where the entire dataset is divided into two subsets which make We split the dataset into training and testing with test size equal to 0.2.

* **Train dataset :** Used to fit the machine learning model.
* **Test dataset :** Used to evaluate the fit machine learning model.



**Fig 5.1 Train and Test split of machine\_status column**

**5.2 Machine Learning models**

There are two approaches to machine learning: supervised and unsupervised. In a supervised model, a training dataset is fed into the classification algorithm.Unsupervised models on the other hand, are fed a dataset that is not labeled and looks for clusters of data points.In our project we are doing the classification model which is a part of supervised learning .

A classification model attempts to draw some conclusion from observed values. Given one or more inputs a classification model will try to predict the value of one or more outcomes.There are a number of classification models. Classification models include logistic regression, decision tree, random forest, gradient-boosted tree, multilayer perceptron, one-vs-rest, and Naive Bayes.

In our project we used different classification models and we just did the comparison of different models using accuracy score,recall ,precision and f1 score as performance metrics.

**Classification Report**

* Precision- Defined as the ratio of true positives to the sum of true and false positives
* Recall-is defined as the ratio of true positives to the sum of true positives and false negatives.
* F1 Score-The F1 is the weighted harmonic mean of precision and recall. The closer the value of the F1 score is to 1.0, the better the expected performance of the model is.
* Support - It is the number of actual occurrences of the class in the dataset. It doesn’t vary between models, it just diagnoses the performance evaluation process.

**Accuracy Score**

Accuracy represents the number of correctly classified data instances over the total number of data instances.

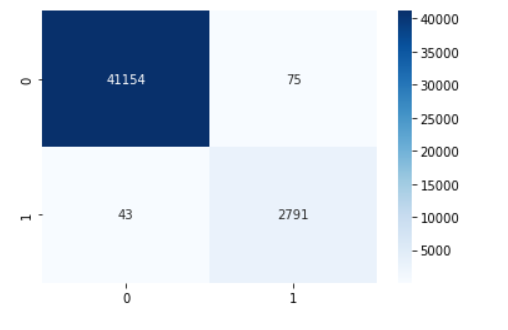
**Confusion Matrix**

It is a matrix of size 2×2 for binary classification with actual values on one axis and predicted on another.

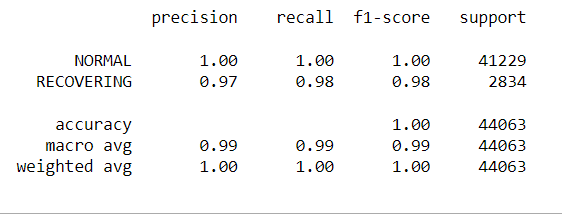
**5.3 Different Classification Models & Classification reports**

**5.3.1 Logistic Regression Model**

It is a supervised learning classification technique that forecasts the likelihood of a target variable. There will only be a choice between two classes. Data can be coded as either one or yes, representing success, or as 0 or no, representing failure. The dependent variable can be predicted most effectively using logistic regression. When the forecast is categorical, such as true or false, yes or no, or a 0 or 1, you can use it.



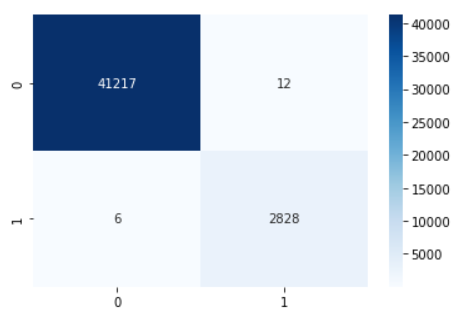
**Fig 5.2 Confusion matrix of Logistic Regression model**



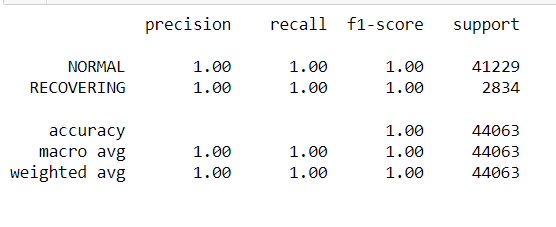
**Fig 5.3 Classification Report of Logistic Regression Model**

**5.3.2 Decision Tree Model**

A decision tree is an example of supervised learning. Although it can solve regression and classification problems, it excels in classification problems. Similar to a flow chart, it divides data points into two similar groups at a time, starting with the "tree trunk" and moving through the "branches" and "leaves" until the categories are more closely related to one another.



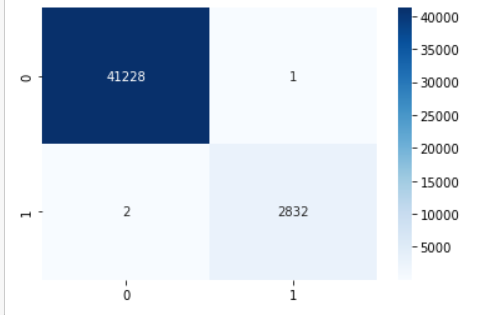
**Fig 5.4 Confusion matrix of Decision Tree model**



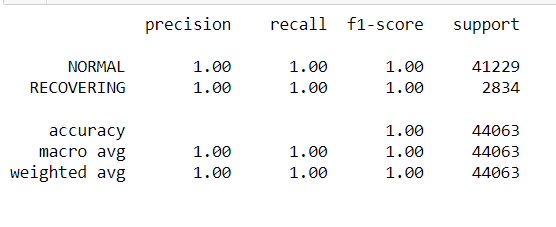
**Fig 5.5 Classification Report of Decision Tree Model**

**5.3.3 Random Forest Algorithm**

The random forest algorithm is an extension of the Decision Tree algorithm where you first create a number of decision trees using training data and then fit your new data into one of the created ‘trees’ as a ‘random forest’. It averages the data to connect it to the nearest tree data based on the data scale. These models are great for improving the decision tree’s problem of forcing data points unnecessarily within a category.



**Fig 5.6 Confusion matrix of Random Forest model**



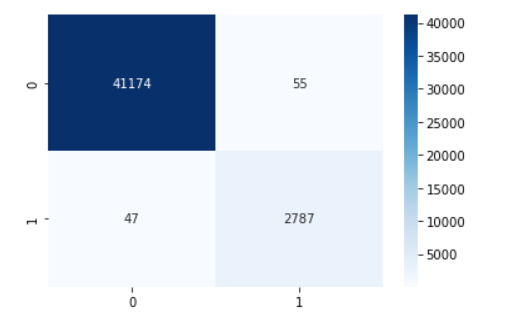
**Fig 5.7 Classification Report of Random Forest Model**

**5.3.4 Support Vector Machine**

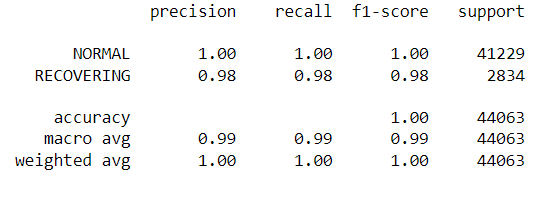
Support Vector Machine is a popular supervised machine learning technique for classification and regression problems. It goes beyond X/Y prediction by using algorithms to classify and train the data according to polarity.

The core idea is to find a maximum hyperplane that best divides the dataset into classes.There are different types of SVM kernels like linear,polynomial ,radial basis function .In our project we used linear and rbf kernels.

* **SVM (linear)**

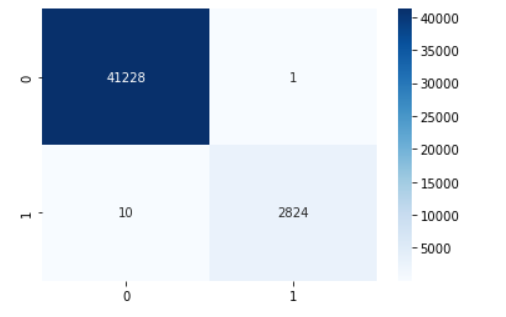
.

**Fig 5.8 Confusion matrix of SVM (linear) model**

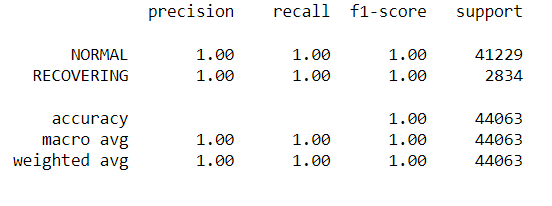


**Fig 5.9 Classification Report of SVM(linear) Model**

* **SVM (rbf)**

****

**Fig 5.10 Confusion matrix of SVM (rbf) model**

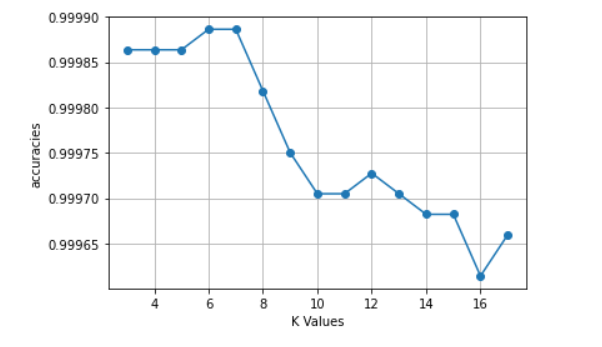
****

**Fig 5.11 Classification Report of SVM(rbf) Model**

**5.3.5 K-Nearest Neighbors**

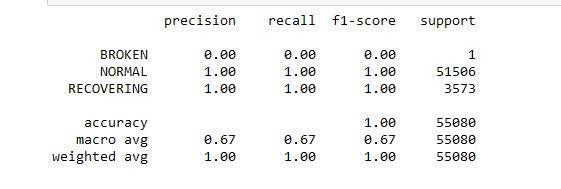
It calculates the likelihood that a data point will join the groups based on which group the data points closest to it are a part of. When using k-NN for classification, you determine how to classify the data according to its nearest neighbor. Steps we followed in knn :

* Choose the number of k of neighbors.
* Take the k nearest neighbor of new data points
* Among the k neighbors, count the number of datapoints in each category
* Assign the new data points to the category where we counted the most neighbors.



**Fig 5.12 K value**

We got maximum accuracy when the K value was equal to 6 ,so we proceeded with the same.

****

**Fig 5.13 Classification Report of knn Model**

**5.4 Comparing Accuracy and F1 Score of Differnet models**

Table 5.1 Comparison of Accuracy and F1 Scores of different models.

| **Model Name** | **Accuracy Score** | **F1 Score : Normal** | **F1 Score : Recovering** |
| --- | --- | --- | --- |
| **Logistic Regression** | 0.9973220162040715 | 0.9985684128794313 | 0.979298245614035 |
| **Decision Tree Classifier** | 0.999659578331026 | 0.99981807817787 | 0.9973558963511369 |
| **Random Forest Classifier** | 0.9999092208882736 | 0.9999514904557472 | 0.9992942836979535 |
| **SVM model** | 0.997685132650977 | 0.9987628865979381 | 0.9820295983086682 |
| **SVM rfb Model** | 0.9997503574427524 | 0.9998666133119916 | 0.9980561936737941 |
| **KNN model** | 0.999886526110342 | 0.9999393638050426 | 0.9991176989588848 |

We can observe from the table that Random Forest Classifier gives us the best results therefore we can select this model considering it has the model with best performance

**Steps Done After selecting the best model :**

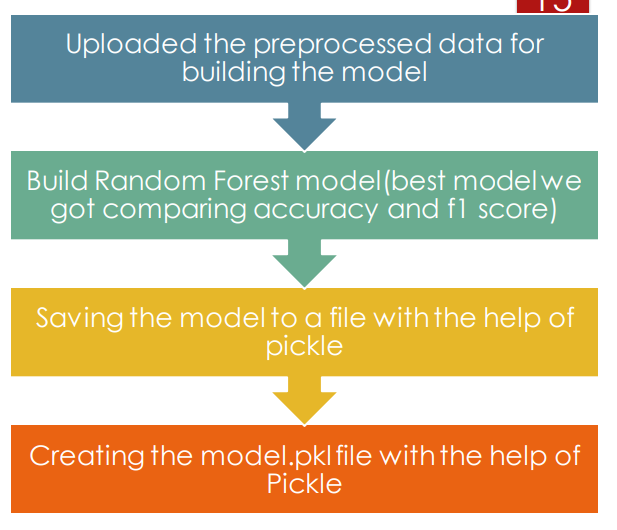
* We are saving the preprocessed file as a seprate csv file with the help of to\_csv command
* This csv file can be used for model creation and futher steps
* So we can eliminate preprocessing steps in model.py file and simplify the proces

**6. WEBSITE HOSTING**

**6.1 Model.py (RANDOM FOREST CLASSIFIER)**

We selected the random forest classifier as the best model considering the accuracy and the f1 score of the model .The first step we have to do in the website hosting is the making of the model.py file .The main function of model.py file is to create model.pkl file which is seriliesed form the model to work in the back end of the page. As mentioned earlier we saved the preprocessed file as a separate csv file which made our job a little easier. Steps followed:

* We uploaded the preprocessed data for building the model as a csv file .
* Then we build the random Forest Model ( best model)
* The next step was the saving of the model with the help of pickle .Pickle in python is primarily used in serializing and deserializing a python object structure.In other words ,it is the process of converting a python object into a byte stream to store it in a file/database .



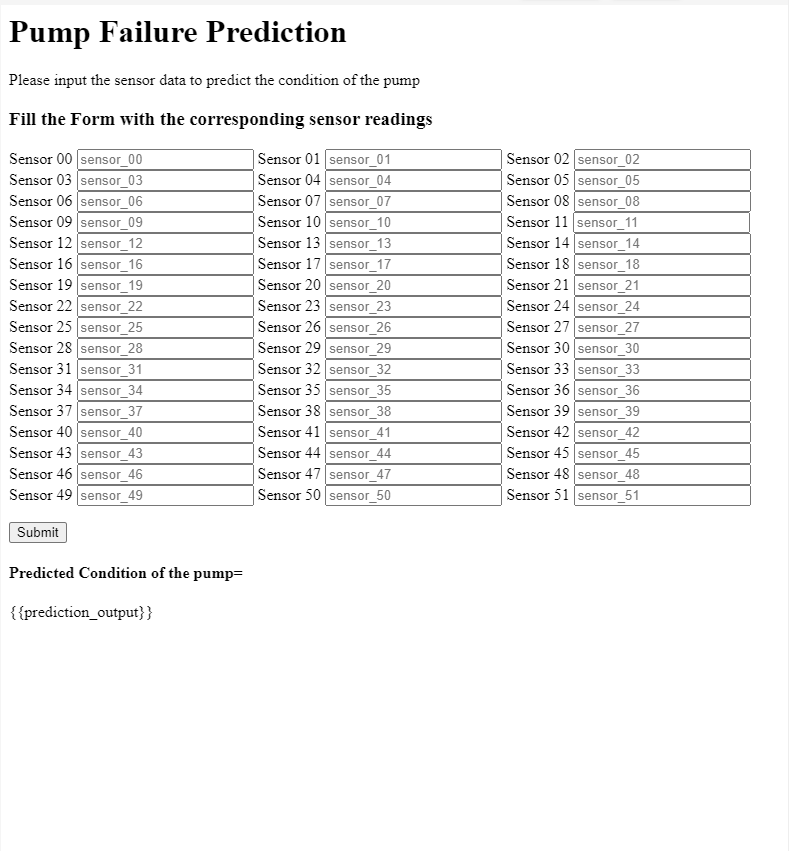
**Fig.6.1 Model.py**

**6.2 Website Making using HTML**

HTML is the standard markup language for Web pages. With HTML you can create your own Website.

. Website is designed in such a way that : .

* Build an Html form to intake the data from the user. We added all the sensor readings to the form.
* The submit button will pass this data to the app.py file.
* This app.py will return value and this will get filled in the region of {{prediction\_output}}.
* Have included some basic tests for better understanding of the Web Page.



**Fig 6.2 HTML page**

The fig 6.2 shows how our webpage HTML looks without any css or any formatting. We can also see the jinga code {{prediciton\_output}} is shown as we added to it. Later this jinga code will be converted to the output with the help of the app.py file which is to be created in the upcoming steps.

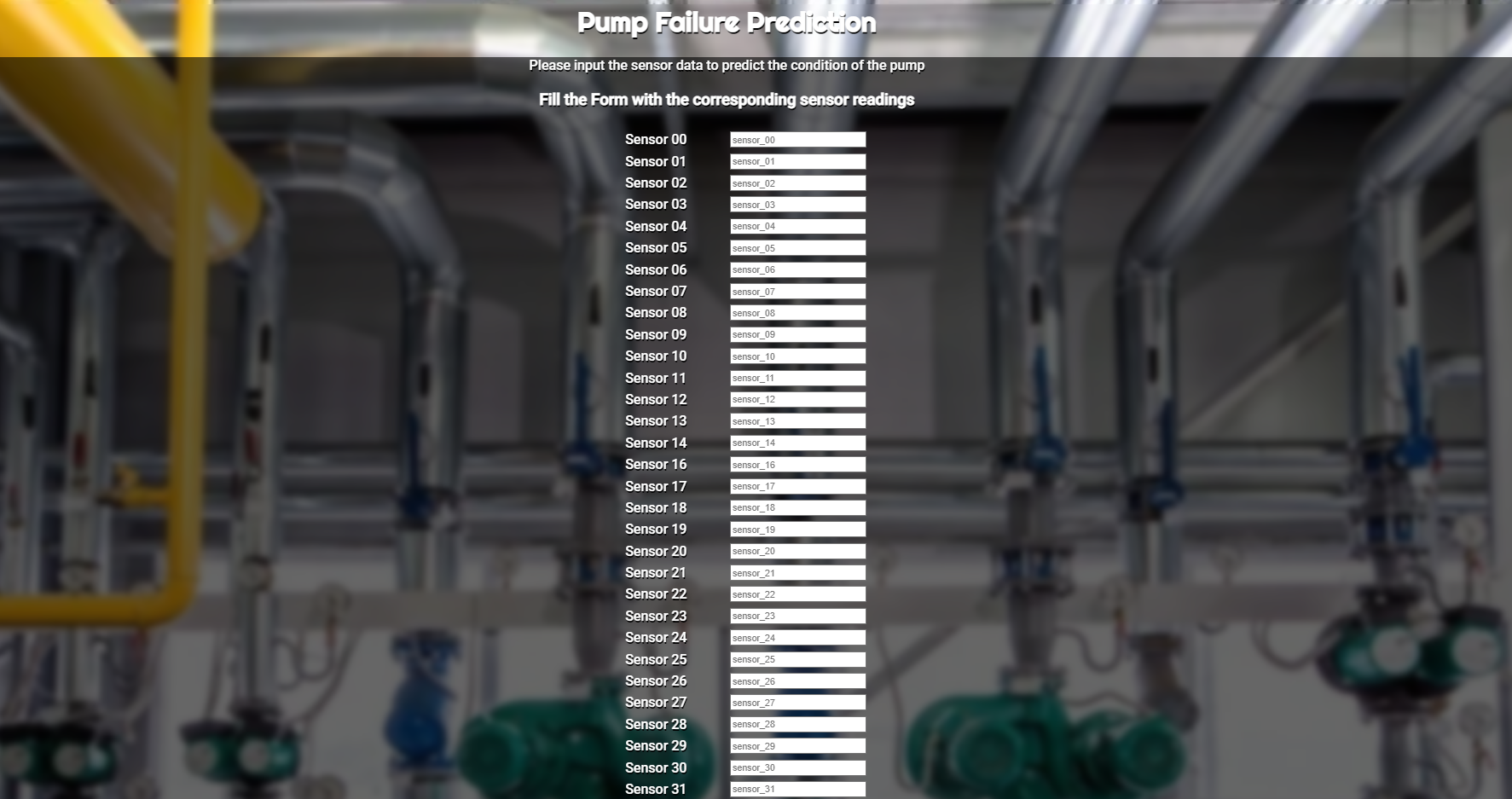
**6.3 CSS**

CSS is used to style an HTML document .CSS describes how HTML elements should be displayed. We can set different formatting colour and backgrounds with the help of CSS

Things we done in css

* Added Background Image
* Included new fonts
* Included a colour overlay
* Changed the formating of submit button
* Seprated background colour overlay for the predcition region.
* We have divided the whole webpage to different sessions and custom formoatting is given to each every sessions of the web page.

CSS made our website more attractive and added aesthetic beauty to our website.

****

**Fig 6.3 Website after doing CSS**

**6.4 Web Application Development using PYTHON FLASK(App.py)**

Flask is a web application framework written in python which is based on the Werkzeug WSGI toolkit and Jinga 2 template engine.It is designed to make getting started quick and easy ,with the ability to scale up to complex applications.

Steps followed in this stage:

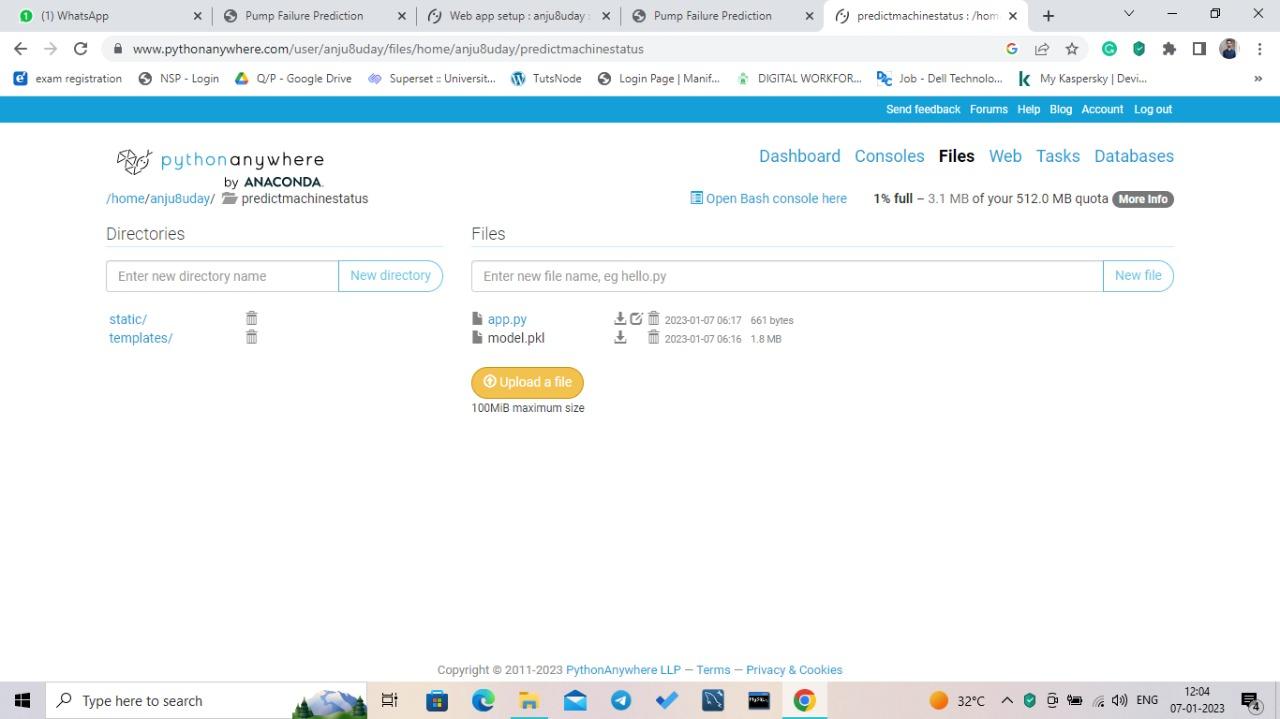
1. We are importing python flask,pickle and numpy library
2. loading model.pkl file which was saved before.
3. We are creating a route url using app.route(‘/’) inorder to render our template in html with the help of render template option
4. Used the requesting method [POST] ,which is the method we specified in the HTML form from where we get the input data.
5. Once the html page is loaded user can give input in the html page and once the enter is pressed after giving the inputs the page get redirected to the predict route as specified in the form hearder
6. Whe predict route created will intake the values from the html note book and convert it int numpy array
7. This numpy array is imputed to the model which was saved before.
8. The model will make the corresponding predictions and the output of it which is in array form is extracted and displayed in the regions of {{predicted\_output}} mention in the web page.

**Local Host Deployment of Web app**

Once the app.py is completed we can make the app.py run. During the running of the app.py file teh terminal will reply as localhost url <http://127.0.0.1:5000/> . We can copy and past this url in the browser inorder to load the webpage. The output was also obtained after we give the necessary input values and submit the html form. This indicate the whole html,css and app.py have been successfully deployed and the web app is ready to be deployed in the server.

**6.5 Web hosting in pythonanywhere**

PythonAnywhere is an online integrated development environment (IDE) and web hosting service (Platform as a service).PythonAnywhere makes it easy to create and run Python programs in the cloud. Python anywhere offers free web app hoisting services and we can easy deploy the web app which we haved made in our system to the web with the help of it. We can write your programs in a web-based editor or just run a console session from any modern web browser.

****

**Fig 6.4 Pythonanywhere**

**Link for the webpage we created:**[**http://anju8uday.pythonanywhere.com**](http://anju8uday.pythonanywhere.com)

Steps we have followed for the web hoisting purpose

1. The app.py,index.html and styles.css are the most important files which are required for the purpose of the web hoisting
2. We are creating a directory named prediction and we are adding folders such as static and templates
3. We are inserting the image for background images and style sheet in the static folder
4. we are inserting the template in the template folder
5. we are inserting the app.py and the model.pkl file to the base directory that is the prediction
6. In this case we have directly inserted the model.pkl file to the python anywhere the main reason for this is that the dataset we are handling with is nearly 200mb and the capacity of python anywhere ins only 100mb
7. so inorder to overcome this limitation we directly uploaded the model.pkl file
8. Once the all the necessary files are uploaded we can go to console region and correct the file location of the files to over files which we have uploaded
9. The next stage is just make the url reload and run.

This is the basic steps to be followed inorder to make our webapp run.

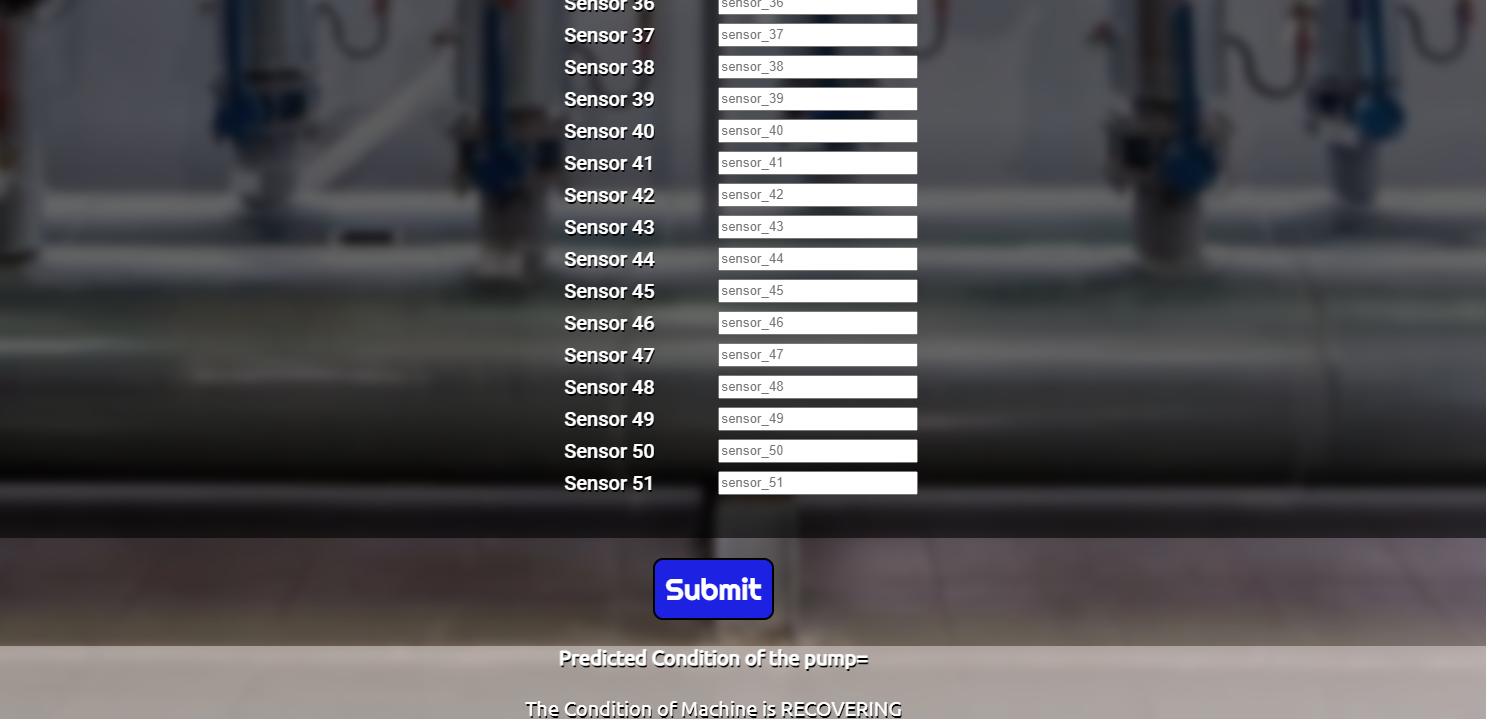
The link of the web application we have made is :-

<http://anju8uday.pythonanywhere.com>

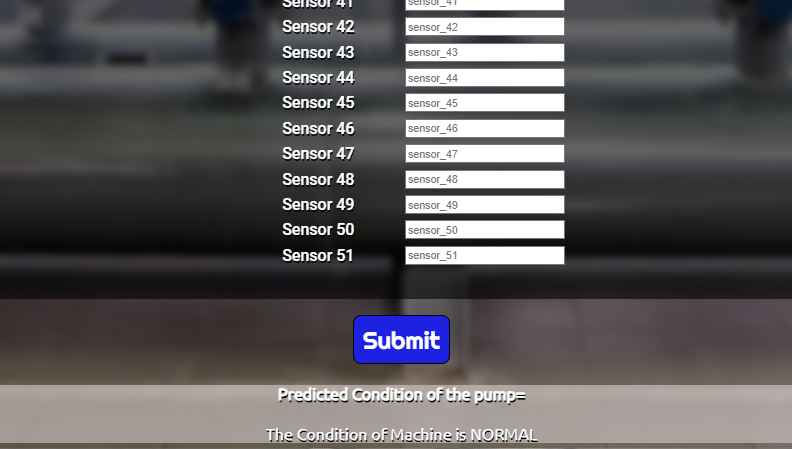
**7. Results**

Predicting the machine condition with the help of a front model which will predict the output with the help of the machine learning model was our objective.

We were able successfully build the machine learning model and link this model to our front that is html and css and make the prediction according to the sensor data that we are giving through the html form. We also successfully built the web app which gives predictions such as NORMAL or Recovering based on the sensor data which we have given.



**Fig 6.5 RECOVERING Condition**



**Fig 6.6 NORMAL Condition**

**8. Conclusion**

The main objective of our project was to make a web app which will be able to predict the machine condition with the help of previous sensor data and we were able to successfully deploy the web app. In this project we have gone through different steps such as data understanding and preprocessing where we totally transformed the data to a format that is best suitable to make a model.

In the next stage we tried different models and found out that the Random forest model which gave the maximum accuracy and F1 score to be our best model and moved forward with it. In this stage we have gone through the implementation and execution of different models which gave us a great practical knowledge. In order to make a good less complex model.py file we planned to extract the preprocessed data from the jupyter notebook and save it as a separate csv file and made use of this extracted csv for the further process. The extracted csv file was used to make model.py file which was meant for making the model.pkl which is a serilased form of the model. We have done only the Random forest model in this case as we came into the insight that the best model is random forest with the help of a series of studies.

The next part of we have made the html form which can intake about 50 sensor readings and submitted. Once the submit button is triggered it will give out the prediction which is the working condition of the pump. Css file is used along with the html inorder to make the html page look better and provide it a better aesthetic look. We were successful in creating html and css files and we were also able to deploy the web app in the local host and make predictions with the help of the model we have created before. The app.py file played a critical role in deploying the web app in the local host.

In the final stage of our project we deployed the webpage in pythonanywhere and we were also able to make successful predictions. Which was the main objective of our project we were able to deploy the same.

**References**

1. Pump sensor data:<https://www.kaggle.com/datasets/nphantawee/pump-sensor-data>
2. Matplotlib (Visualization)<https://matplotlib.org/>
3. Numpy:<https://numpy.org/>
4. LabelEncoding:<http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html>
5. Machine Learning: <https://scikit-learn.org/stable/>
6. FeatureEngineering:<https://www.analyticsvidhya.com/blog/2021/04/guide-for-feature-extraction-techniques/>
7. ClassificationModels:<https://www.analyticsvidhya.com/blog/2020/11/popular-classification-models-for-machine-learning/>
8. ClassificationReport:<https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html>
9. HTML &CSS:<https://www.w3schools.com/html/>
10. OutlierHandling:<https://www.analyticsvidhya.com/blog/2021/05/detecting-and-treating-outliers-treating-the-odd-one-out/>
11. Pythonanywhere:<https://www.pythonanywhere.com/>
12. Scaling:<https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/>