Epileptic Disorder Prediction

1 Technical Design Document

Version 1.0

RESTRICTED DISTRIBUTION

The information is standard Company Confidential, but due to its sensitivity, it has restricted distribution and viewing within iNeuron.

Document Version Control

Date Issued	Version	Description	Author
13th August 2020	1.0	Initial Draft	Sanket Mote
14th August 2020	2.0	Reviewd & Updated	Jyotsna Singh
17th August 2020	3.0	Updated the deployment Section	Jyotsna Singh
19th August 2020	4.0	Updated Testing Modules & Monitoring Dashboard section as per feedback from Virat & Mohit	Jyotsna Singh
20th August 2020	5.0	Updates the Deployment section for Dockerization details	Jyotsna Singh

Contributors

The content of this document has been authored with the combined input of the following group of key individuals.

Name	Section Worked Upon
Sanket Mote	Initial Draft
Jyotsna Singh	Initial Draft (Technical Inputs)

Document Classification

Classification	Company Confidential
Definition	Information is Group confidential and needs to be protected
Context	Where the loss of information confidentiality would result in significant harm to the interests of the organisation, financial loss, embarrassment or loss of information

Contents

1. Introduction

The goal here is to build an end to end automated Machine Learning solution where the user will give the data, and the result will be the prediction of epileptic disorder based on prediction generated using hyper tuned Machine Learning model.

This project shall be delivered in two phases:

Phase 1: All the functionalities with PyPi packages.

Phase2: Integration of UI to all the functionalities.

The technical design document gives a design blueprint of the Epileptic Disorder disease prediction project. This document communicates the technical details of the solution proposed.

In addition, this document also captures the different workflows involved to build the solution, exceptions in the workflows and any assumptions that have been considered.

Once agreed as the basis for the building of the project, the flowchart and assumptions will be used as a platform from which the solution will be designed.

Changes to this business process may constitute a request for change and will be subject to the agreed agility program change procedures.

Note: All the code will be written in python version 3.6

1.1 High level objectives

The high-level objectives are:

1. Enable reading/loading of data from the csv file format as input and convert them into pandas dataframe.
2. Perform statistical analytics of the data and prepare a report for the analysis.
3. Perform graphical analysis for the data.
4. Perform basic data validation steps
5. Perform data cleaning operation with all the steps required and preprocess the data for training or prediction.
6. After data cleaning save the preprocessed files in the file system.

If Prediction

7. Load the appropriate best ML model for prediction or training.
8. Perfom prediction and display/download prediction results based on single or bulk upload.

If Re-Training

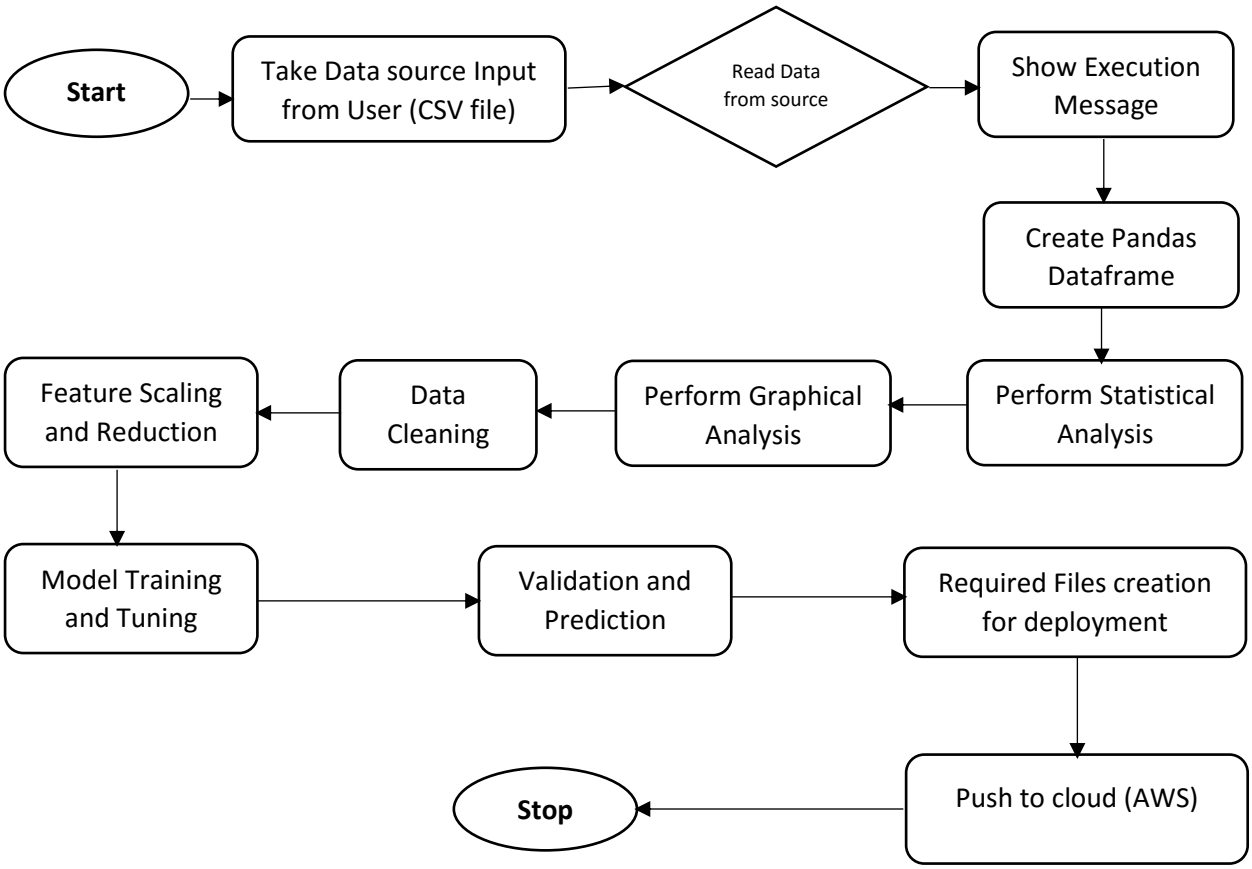
9. Perform Re-train based on the best model selected with tuned hyper parameter
10. Create the multiple metrics for ML model incase of training to evaluate model.
11. Cloud deployment of the model on AWS so that end user can access it.

Phase 1: Create Pypi packages

Phase 2: Create UI

2 Workflow Overall

2.1 Application Flow



2.2 Exception Scenarios Overall

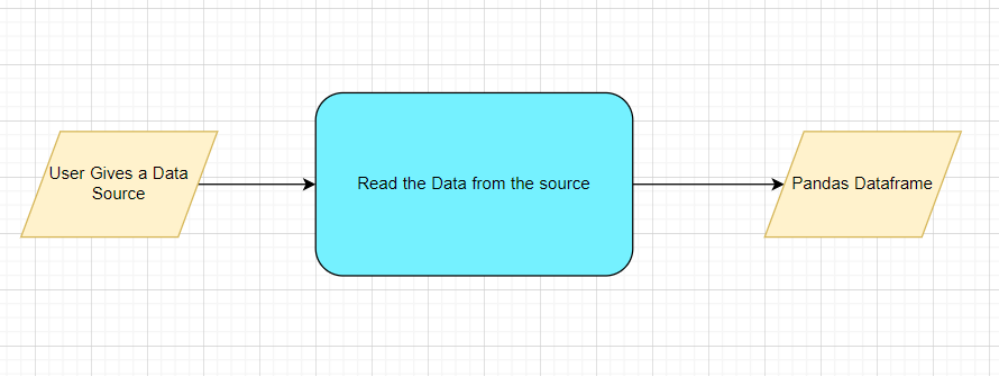
Step	Exception	Mitigation
User gives Wrong Data Source	Give proper error message	Ask the user to re-enter the details
User gives corrupted data	Give proper error message	
User gives wrong null symbol	Give proper error message	Ask the user to provide correct symbol used for missing values
Deployment credentials are wrong	Give proper error message	Ask for the details to be entered again

3 Workflow Data Ingestion and File Conversion

Data Sources:

Phase 1:

3.1 Technical solution design



3.2 Method Definitions

Class Name	DataGetter	
Method Name	upload_file	
	Method Description	This method will be used to upload the file provided by the user for Prediction or Re-Training
	Input parameter names	self, file
	Input Parameter Description	file: it will contain the input csv file from the request object that has been upload by the user
	Output	A pandas Dataframe
	On Exception	UploadFile_Log Write the exception in the log file. Raise an exception with the appropriate error message
Method Name	data_validation	
	Method Description	This method will be used to read data from a csv file, after loading validate if file has correct number of columns, data types of the features are as per problem statement.
	Input parameter names	self, file_path_val, process_type
	Input Parameter Description	file_path: path of the file from where it can be accessed and loaded for validation process_type: this will help the program to identify whether the uploaded data is for training or prediction
	Output	A pandas Dataframe
	On Exception	DataValidation_Log Write the exception in the log file. Raise an exception with the appropriate error message
Method Name	data_preprocess	
	Method Description	This method will be used to read data from a validated and saved csv file.
	Input parameter names	self, file_path_val, process_type

	Input Parameter Description	file_path: path of the file from where it can be accessed and loaded for pre-processing process_type: this will help the program to identify whether the uploaded data is for training or prediction Above function preprocesses data(removed redundant columns), scales down the data using standard_scalar and transforms data using PCA
	output	A pandas Dataframe
	On Exception	PreProcessing_Log Write the exception in the log file. Raise an exception with the appropriate error message
Method Name	predict_model	
	Method Description	This method will be used to do predictions for the dataset provided
	output	A pandas Dataframe, output to the html page for end user to review A CSV file with prediction results stored in the file system
	On Exception	Prediction_Log Write the exception in the log file. Raise an exception with the appropriate error message
Method Name	train_model	
	Method Description	This method will be used to re-train the model
	ouptput	A serialized model file in (.sav) format
	On Exception	Write the exception in the log file. Raise an exception with the appropriate error message
Method Name	file_downloads	
	Method Description	This method will be used to provide option to user to download the Results file with prediction results

	ouptput	A CSV file with prediction results with an option to download and save on system
	On Exception	Raise an exception with the appropriate error message

3.3 Exceptions Scenarios

Step	Exception	Mitigation
User gives Wrong Data Source	Give proper error message	Ask the user to re-enter the details
User gives corrupted data	Give proper error message	Ask the user to re-enter the details

4 Stats Based EDA

4.1 Steps

Distance Plot to show data distribution wrt to mean & SD for rows & columns

VIF - Variance Inflation Factore

Correlation coefficient Matrix

Data scaling using Standard Scalar

PCA - Principal Component Analysis

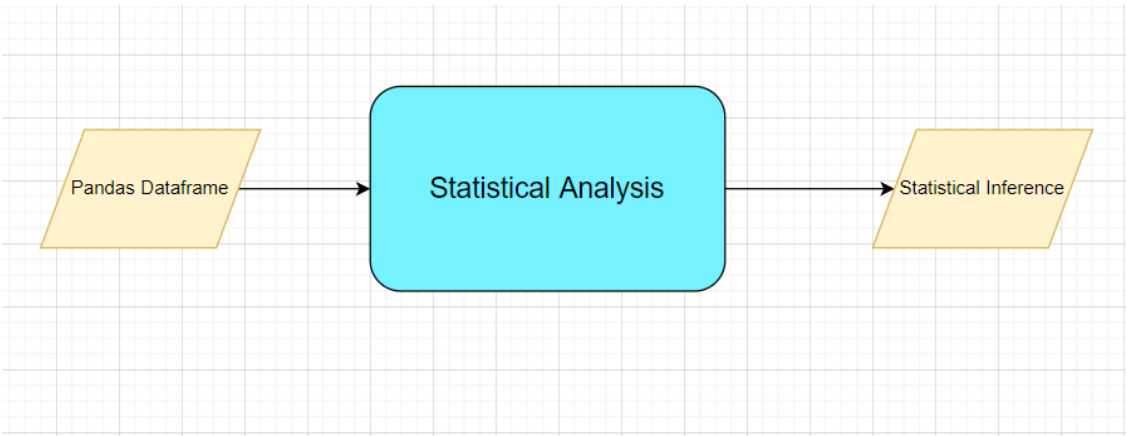
Column contributions/ importance

SMOTE & Weights for Class Balancing

SweetViz baded EDA

Box Plot for outlier detection

4.2 Technical solution design



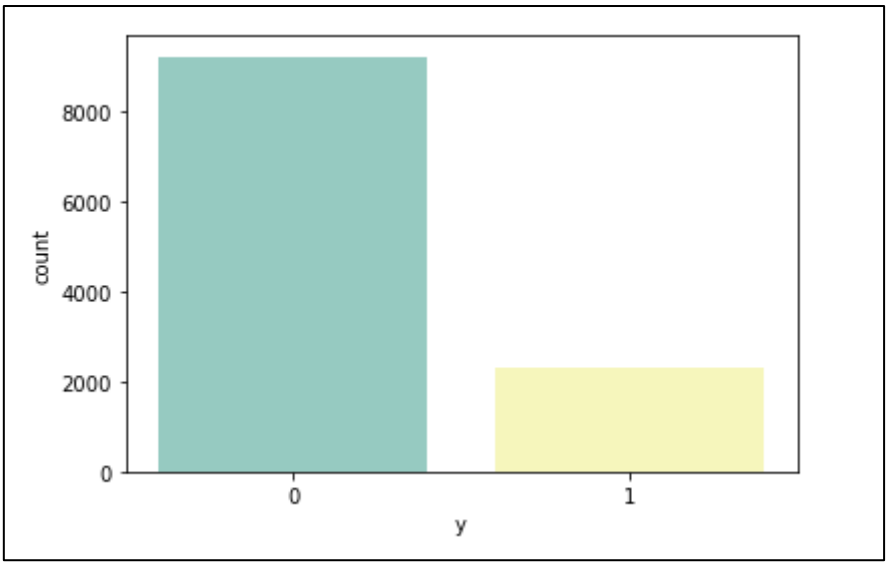
4.3 Exceptions Scenarios Module Wise

Step	Exception	Mitigation
Column has mixed values(Integer & number)	Give proper error message	Ask the user to correct the data.

5 Graph-Based EDA

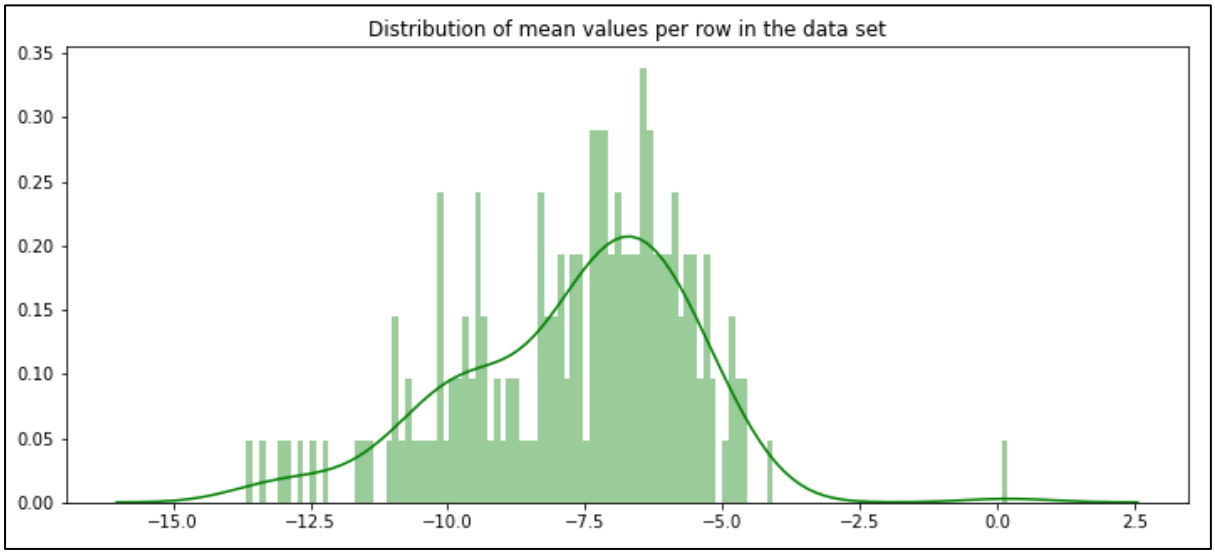
Create the following graphs:

Check for balance/imbalance



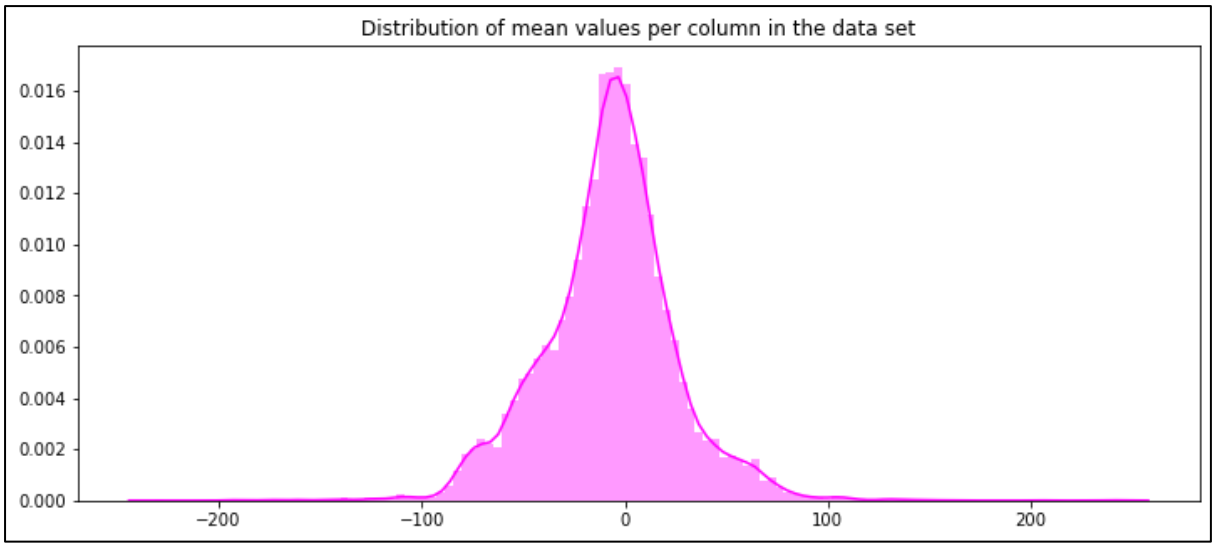
Inference – Class imbalance is present with about 80:20 ratio for Class 0:Class 1

Mean distribution between rows



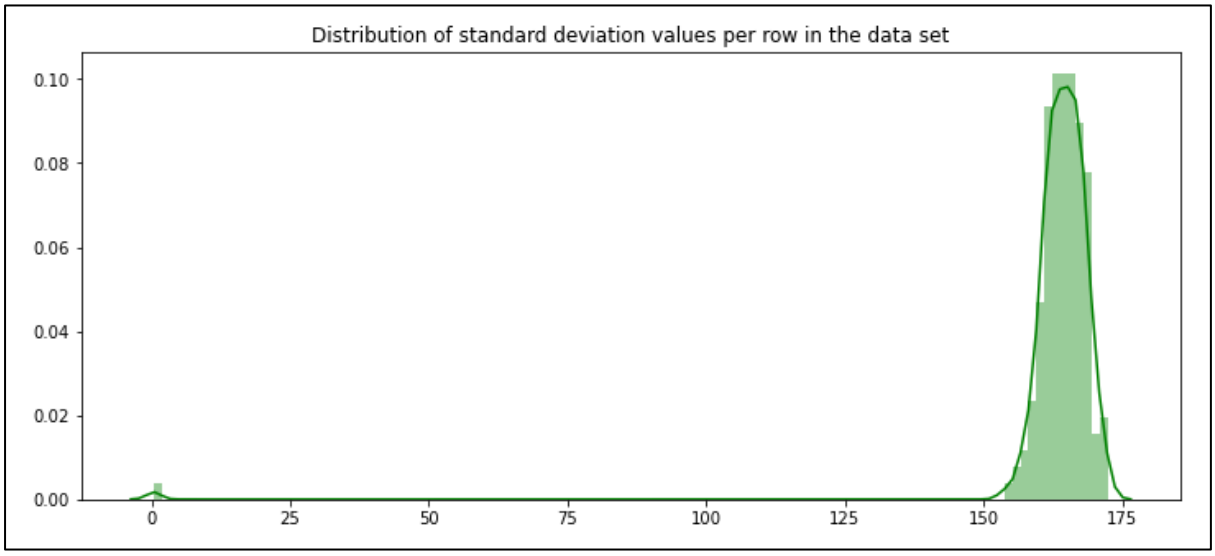
Inference – Data is somewhat normally distributed in terms of mean values per rows in the dataset, it is not completely skewed

Mean distribution between columns



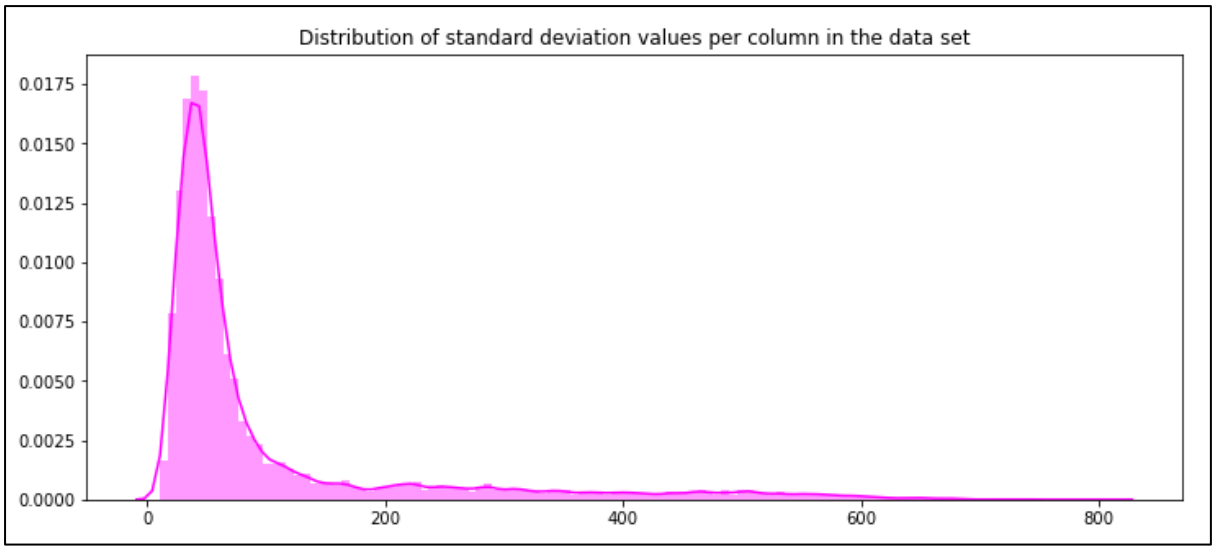
Inference - Data is normally distributed in terms of mean values per columns in the dataset

Standard Deviation distribution between rows



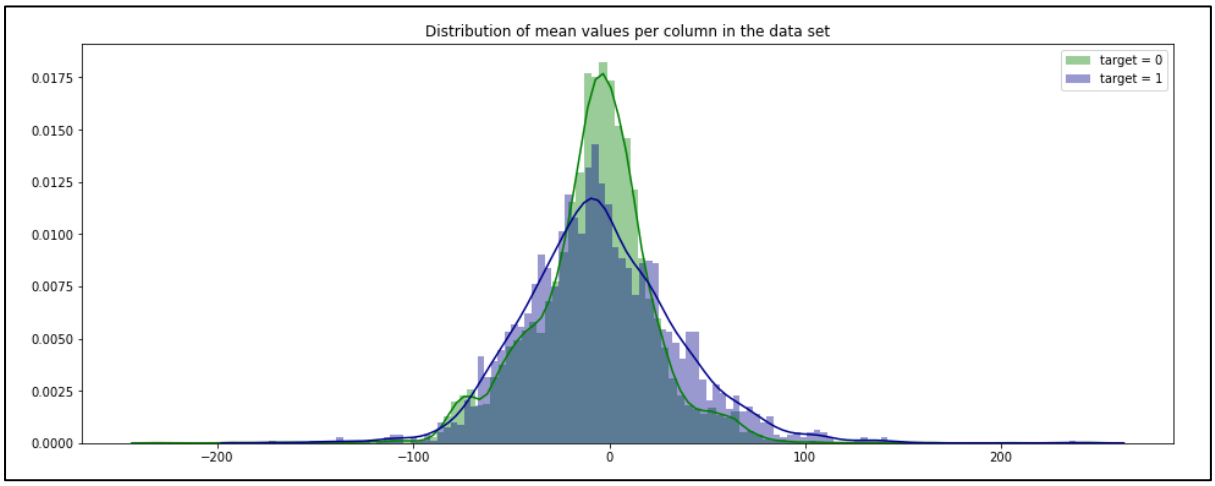
Inference - Data is left skewed with respect to standard deviation per row in the dataset

Standard Deviation distribution between columns



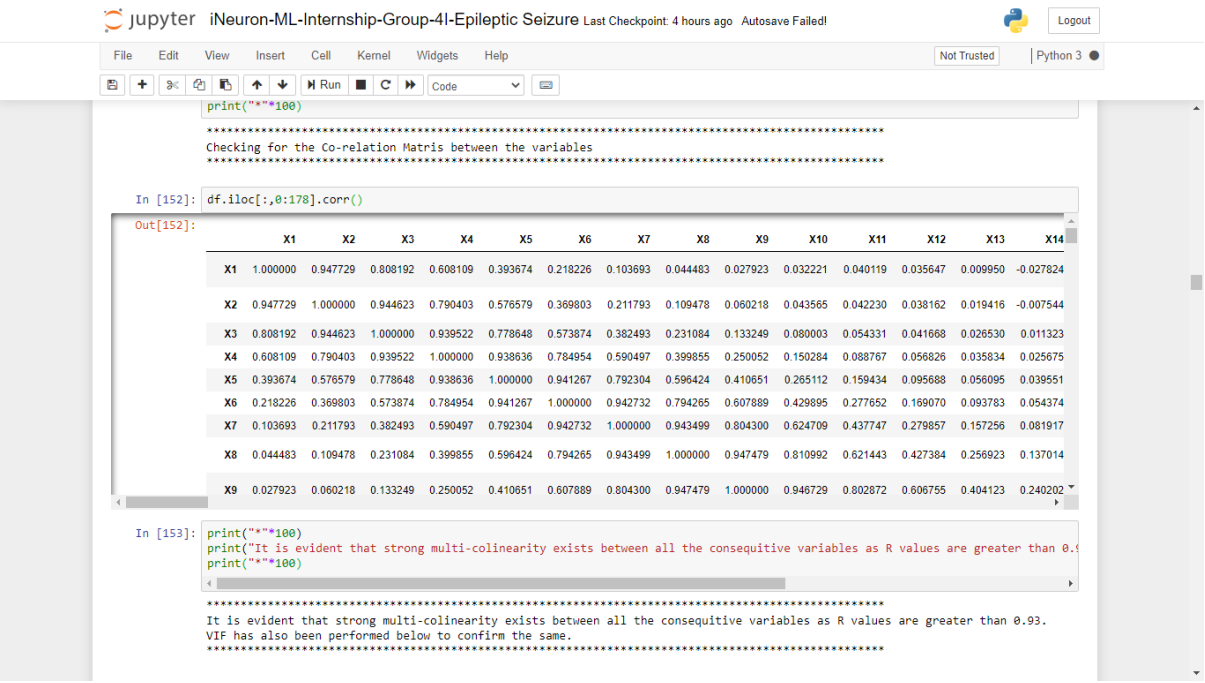
Inference - Data is right skewed with respect to standard deviation per column in the dataset

Majority and Minority distribution wise comparison



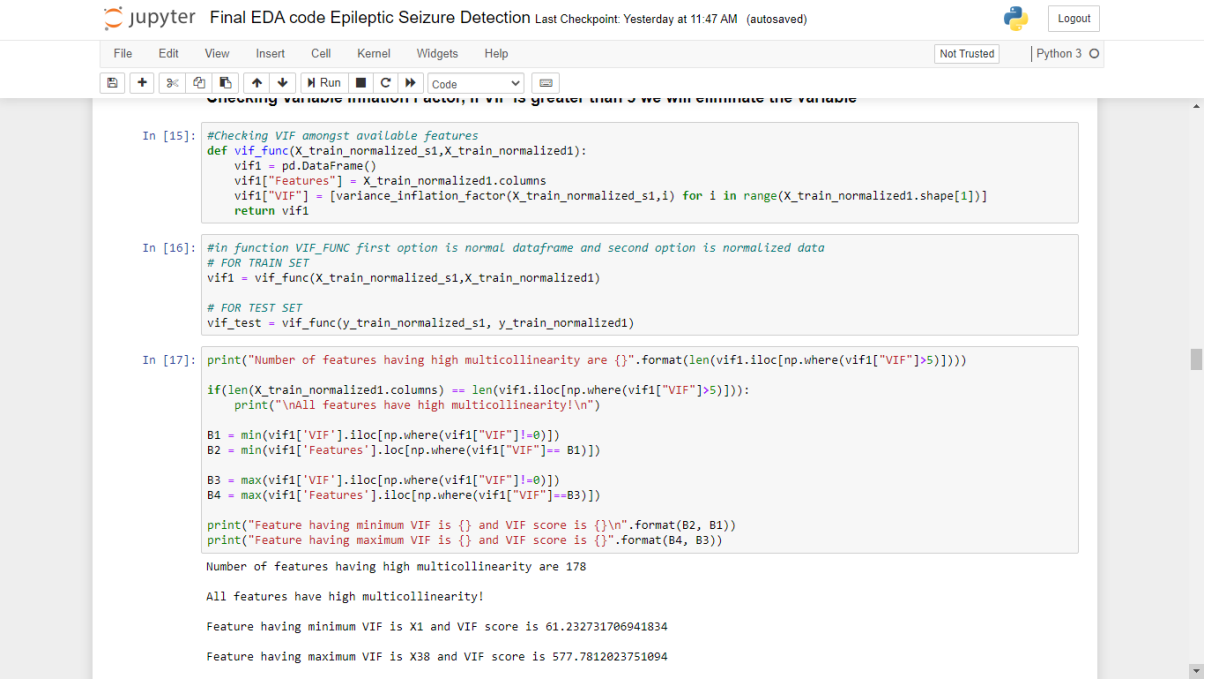
Inference - Data distribution with respect to dependent variable in the majority and minority class seems to be evenly spread with respect to the mean in the dataset

Correlation Coefficient Matrix

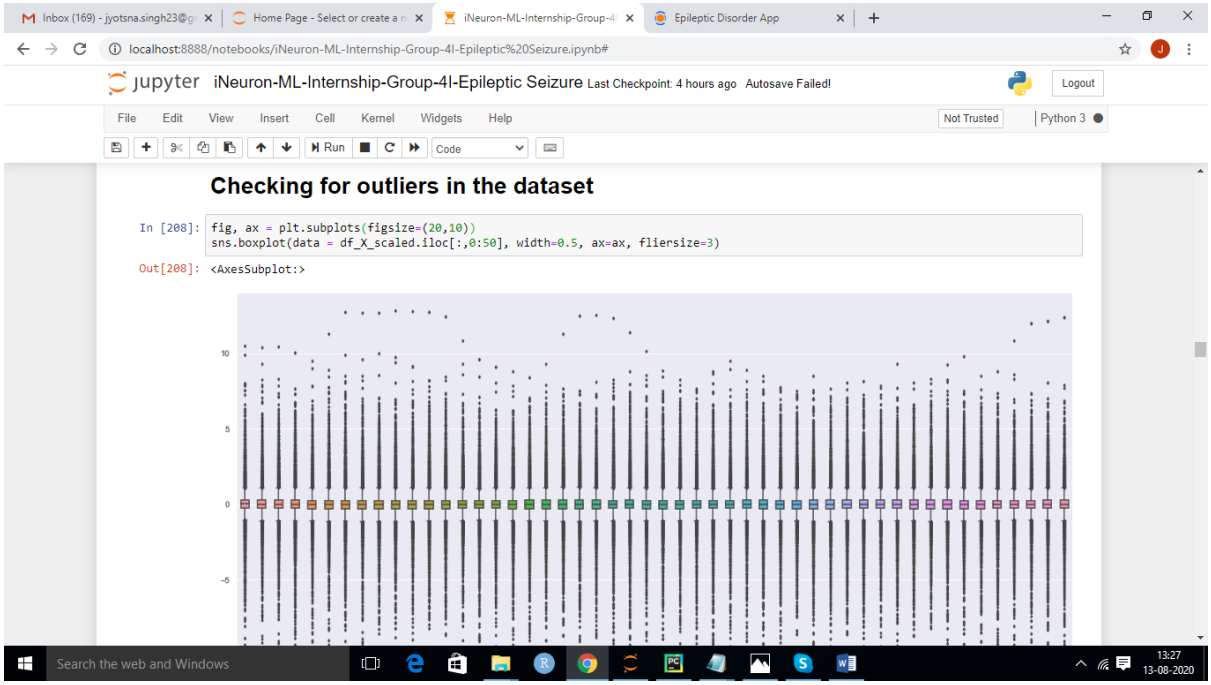


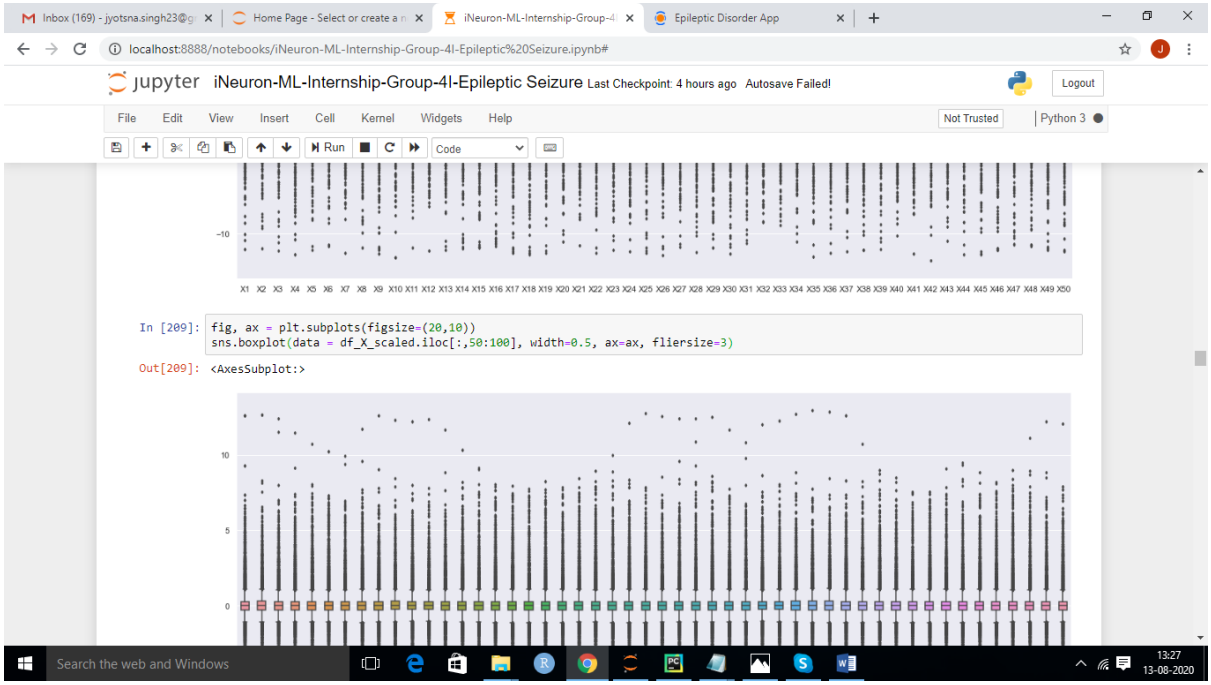
Inference - It is evident that strong multi-colinearity exists between all the consequitive variables as R values are greater than 0.93. VIF has also been performed below to confirm the same.

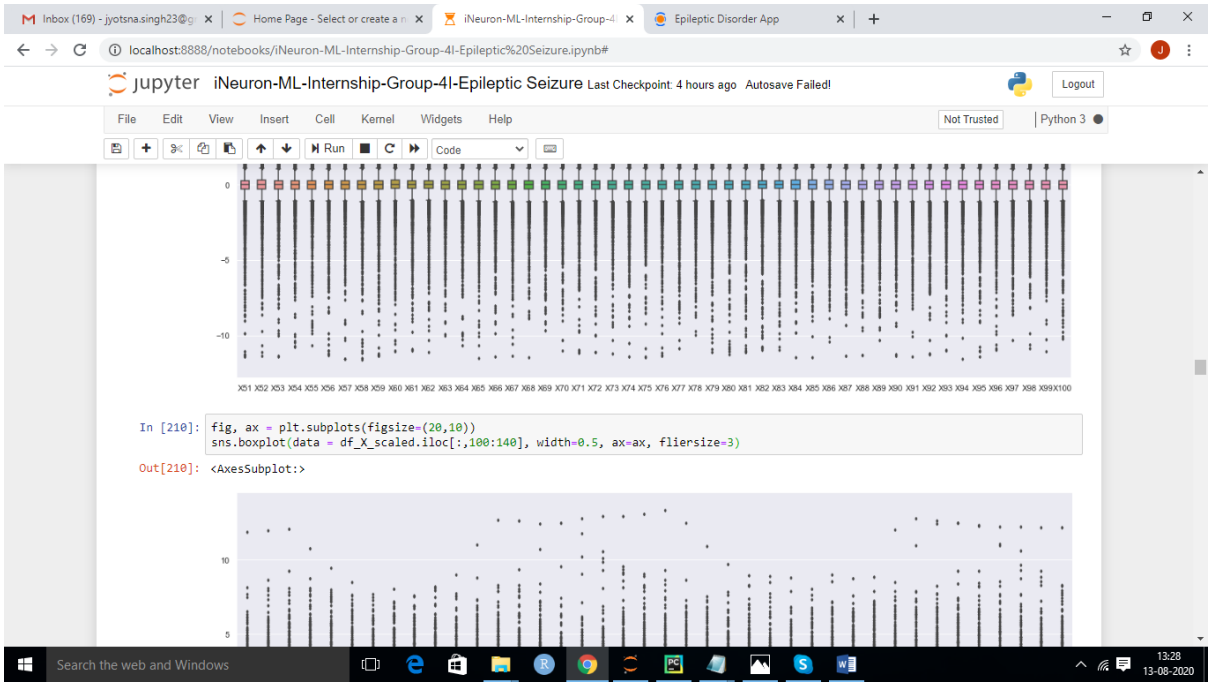
VIF Check for Features

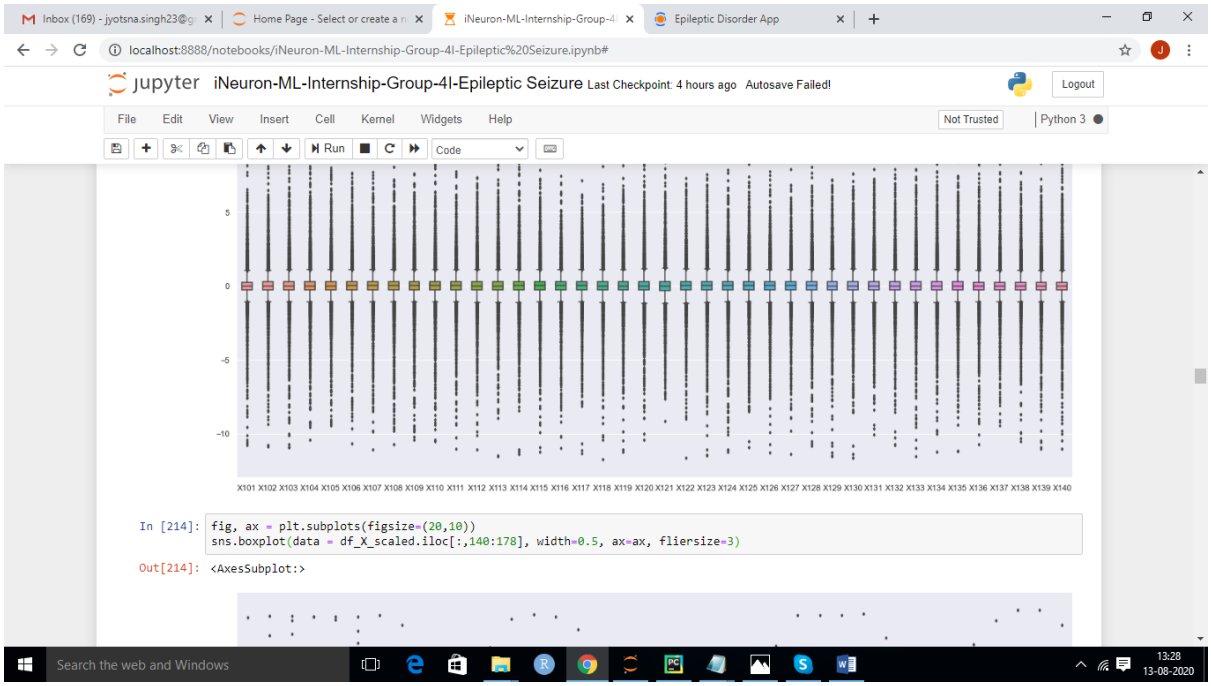


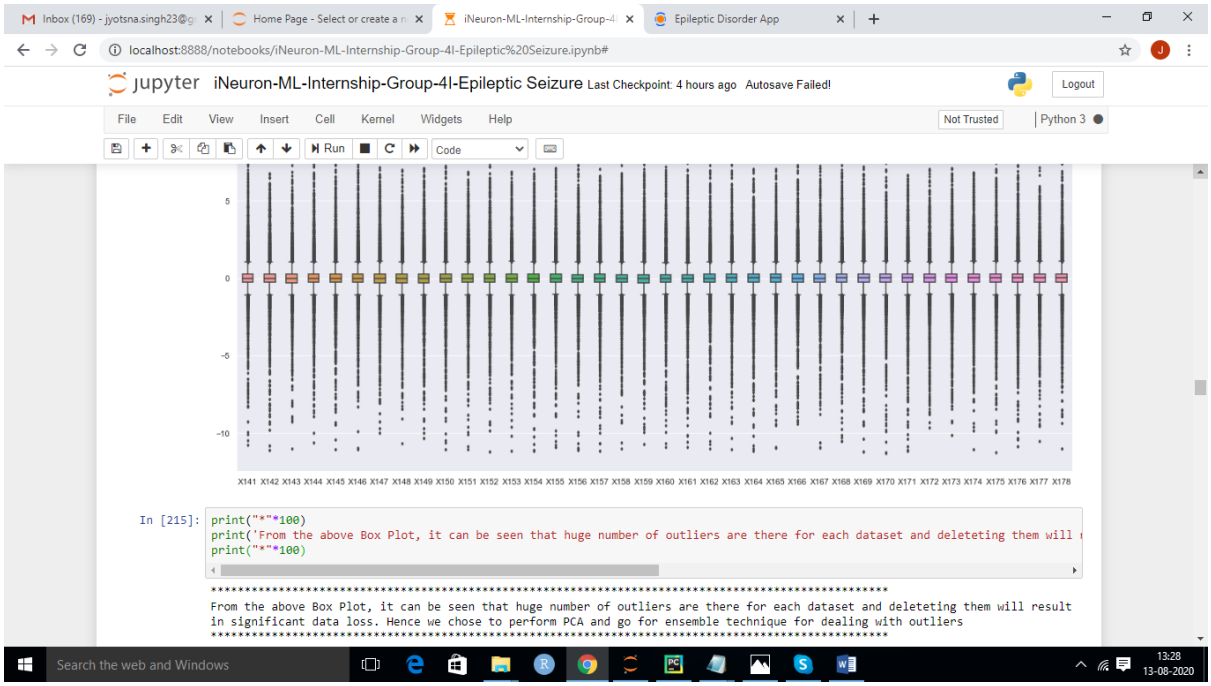
Outlier Detection using BOX Plot






Inbox (169) - jyotsna.singh23@g
Home Page - Select or create a n
iNeuron-ML-Internship-Group-4I
Epileptic Disorder App
localhost:8888/notebooks/iNeuron-ML-Internship-Group-4I-Epileptic%20Seizure.ipynb#
jupyter iNeuron-ML-Internship-Group-4I-Epileptic Seizure Last Checkpoint: 4 hours ago Autosave Failed!
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code
0
−5
−10
X51 X52 X53 X54 X55 X56 X57 X58 X59 X60 X61 X62 X63 X64 X65 X66 X67 X68 X69 X70 X71 X72 X73 X74 X75 X76 X77 X78 X79 X80 X81 X82 X83 X84 X85 X86 X87 X88 X89 X90 X91 X92 X93 X94 X95 X96 X97 X98 X99X100
In [210]:
fig, ax = plt.subplots(figsize=(20,10))
sns.boxplot(data = df_X_scaled.iloc[:,100:140], width=0.5, ax=ax, fliersize=3)
Out[210]:
<AxesSubplot:>
10
5
Search the web and Windows
13:28
13-08-2020


Inbox (169) - jyotsna.singh23@g
Home Page - Select or create a n
iNeuron-ML-Internship-Group-4I
Epileptic Disorder App
localhost:8888/notebooks/iNeuron-ML-Internship-Group-4I-Epileptic%20Seizure.ipynb#
jupyter iNeuron-ML-Internship-Group-4I-Epileptic Seizure Last Checkpoint: 4 hours ago Autosave Failed!
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code
5
0
−5
−10
X101 X102 X103 X104 X105 X106 X107 X108 X109 X110 X111 X112 X113 X114 X115 X116 X117 X118 X119 X120 X121 X122 X123 X124 X125 X126 X127 X128 X129 X130 X131 X132 X133 X134 X135 X136 X137 X138 X139 X140
In [214]:
fig, ax = plt.subplots(figsize=(20,10))
sns.boxplot(data = df_X_scaled.iloc[:,140:178], width=0.5, ax=ax, fliersize=3)
Out[214]:
<AxesSubplot:>
Search the web and Windows
13:28
13-08-2020

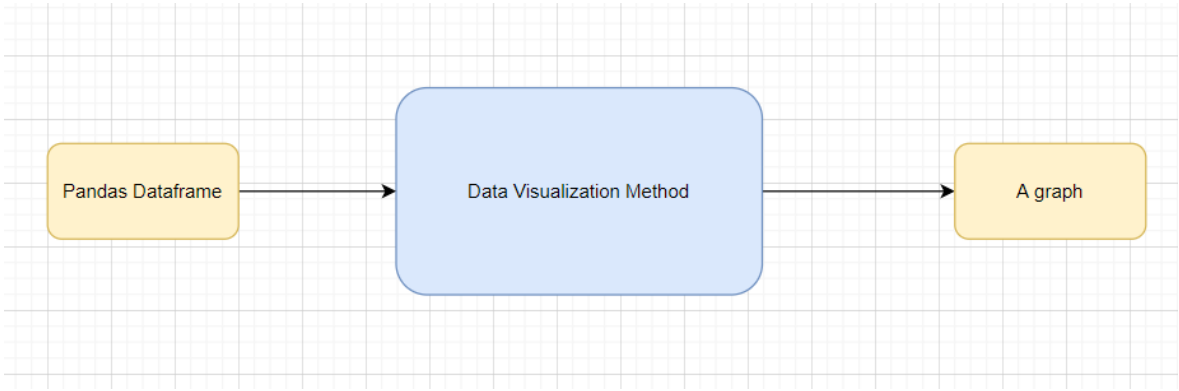


From the above Box Plot, it can be seen that huge number of outliers are there for each dataset and deleteting them will result in significant data loss. Hence have performed PCA to deal with outliers

Anaysis from **SweetViz – Report.html** is also available

Basis Analysis using **Plotly** was also done.

5.1 Technical solution design



6 Data Transformers(Pre-processing steps)

MVP:

Null value check

Data type check

Imbalanced data set handling

Standardization using standard scals

Data Transformation using PCA

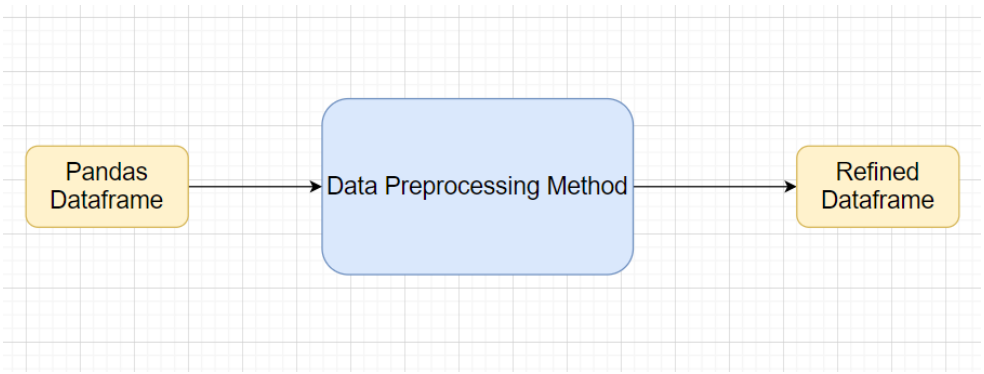
Phase1:

Outlier detection

Data Scaling

Feature Selection using SelectKBest: https://scikit-learn.org/stable/auto_examples/index.html#feature-selection

6.1 Technical solution design



6.2 Exceptions Scenarios Module Wise

Step	Exception	Mitigation
Wrong parameters passed to the methods	Handle Internally	Code should never give a wrong input

7 ML Model Selection & Optimization

Note: The data should have been divided into train and validation set before this.

MVP:

Following Classification algorithms were run

1) Logoistics Regression Classifier
2) Decision Tree Classifier
3) Random Forest Classifier
4) Ada Boost Classifier
5) Support Vector Classifier
6) K-Nearest Neighbour Classifier
7) Naïve Bayes

Logistics Regression, Decision Tee and Naïve Bayes gave unsatisfactory results and hence were dropped and remaining 4 classifiers were further taken to improve the performance on below criterais

1) Default parameters
2) Hyper parameter tuning and models with tuned hyper parameters
3) Feature Selections
4) AUTO ML – Using TPOT classifier

AUTO ML – TPOT

Used TPOT which identified the best pipeline which was KNN Classifier.

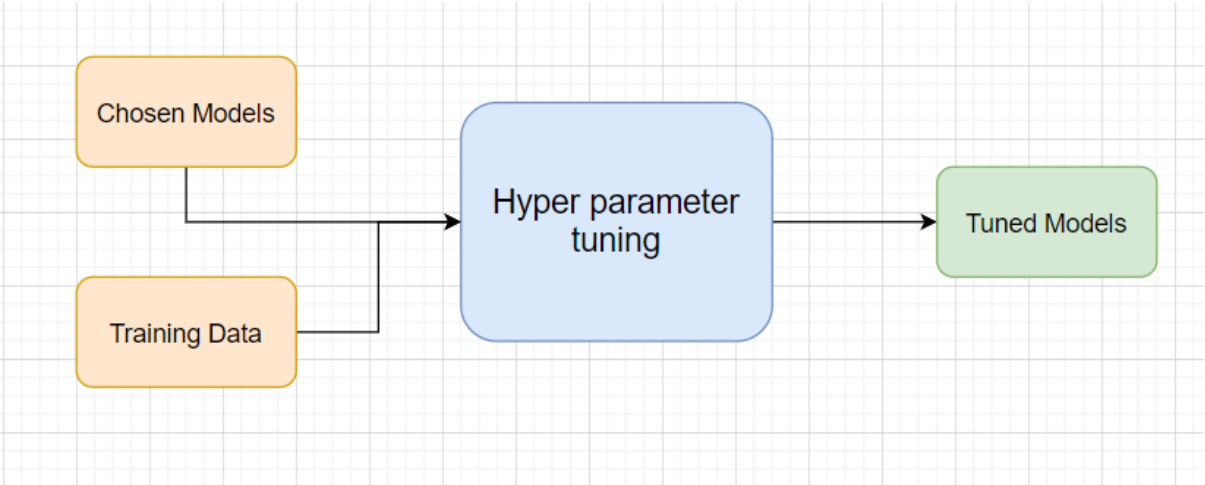
Phase1:

Model Selection criteria – Recall, Accuracy, F1 Score, Precision

Classifier		F1 Score	Precision	Recall	Accuracy
RFC-def	Class 0	0.98	0.96	1.00	0.96
	Class1	0.92	0.99	0.85	
RFC-hyp	Class 0	0.98	0.97	0.99	0.97
	Class1	0.93	0.98	0.89	
ABC-def	Class 0	0.97	0.96	0.98	0.95
	Class1	0.89	0.94	0.84	
ABC-hyp	Class 0	0.98	0.98	0.99	0.97
	Class1	0.94	0.97	0.91	
SVC-def	Class 0	0.98	0.98	0.99	0.98
	Class1	0.94	0.97	0.91	
SVC-hyp	Class 0	0.99	0.99	0.99	0.98
	Class1	0.94	0.94	0.94	
KNN-def	Class 0	0.98	0.99	0.96	0.96
	Class1	0.89	0.84	0.96	
KNN-hyp	Class 0	0.96	1.00	0.94	0.94
	Class1	0.82	0.71	0.97	
TPOT-KNN	Class 0	0.94	0.99	0.96	0.96
	Class1	0.98	0.84	0.96	

Based on the above results of **Recall, Accuracy & F1 Score** **Support Vector Classifier with tuned hyper parameter** was identified as the final model to be used for Predictions.

7.1 Technical solution design



7.2 Exceptions Scenarios Module Wise

Step	Exception	Mitigation
Wrong parameters passed to the methods	Handle Internally	Code should never give a wrong input

8 Testing Modules

Divide the training data itself into train and test sets

Use test data to have tests run on the four best models (RFC, AdaBoost, KNN, SVC)

Give the test report based on the below scores.

a) F1 Score
b) Accuracy
c) Precision
d) Recall

From these scores, most important score to select the best Model in the order of their importance are listed below :-

1) **Recall –** Since this is a medical disorder prediction scenario and Sensitivity is most important score. Model should not predict ‘No Disorder’ for a patient having ‘A disorder’. If model does that then it will be a disaster. Hence Good Recall is most important in this project
2) **F1 Score –** As it is a harmonic mean of Precision & Recall, this performance metrics is also important for the decision on the selection of best model
3) **Accuracy –** This also is important for overall model’s performance

9 Prediction Pipeline

Use the existing data read modules

Use the existing data validation modules

Use the existing pre-processing module

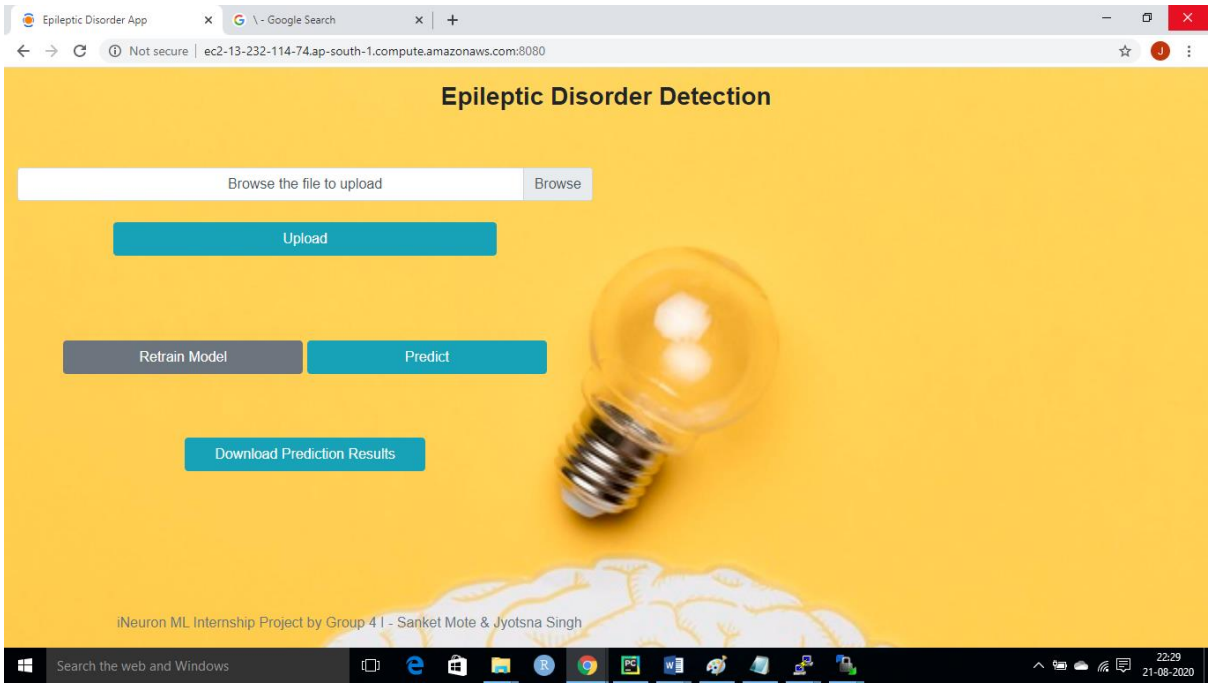
Load the model into memory

Do predictions

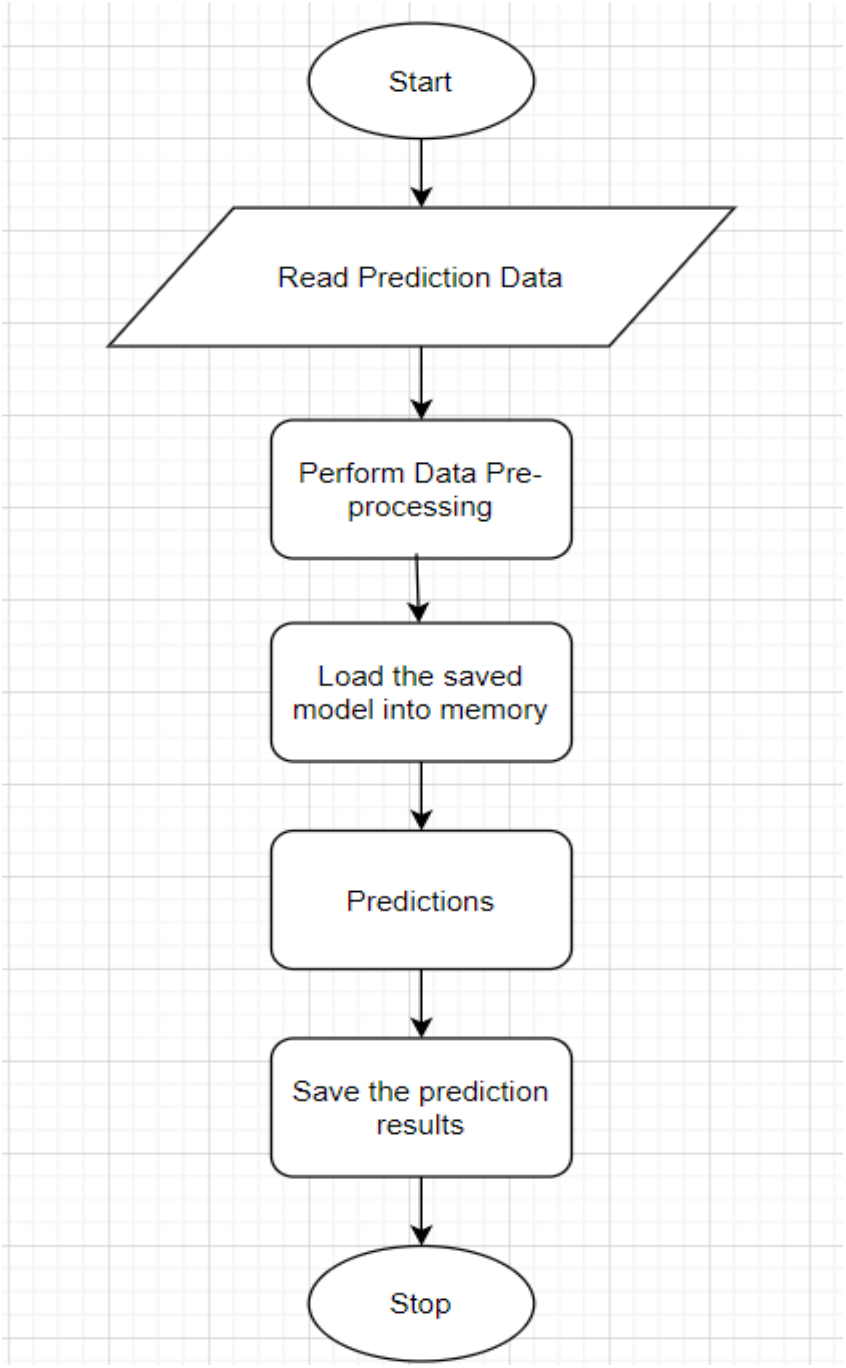
Store & display the prediction results

Phase 2:

UI for predictions



9.1 Technical solution design

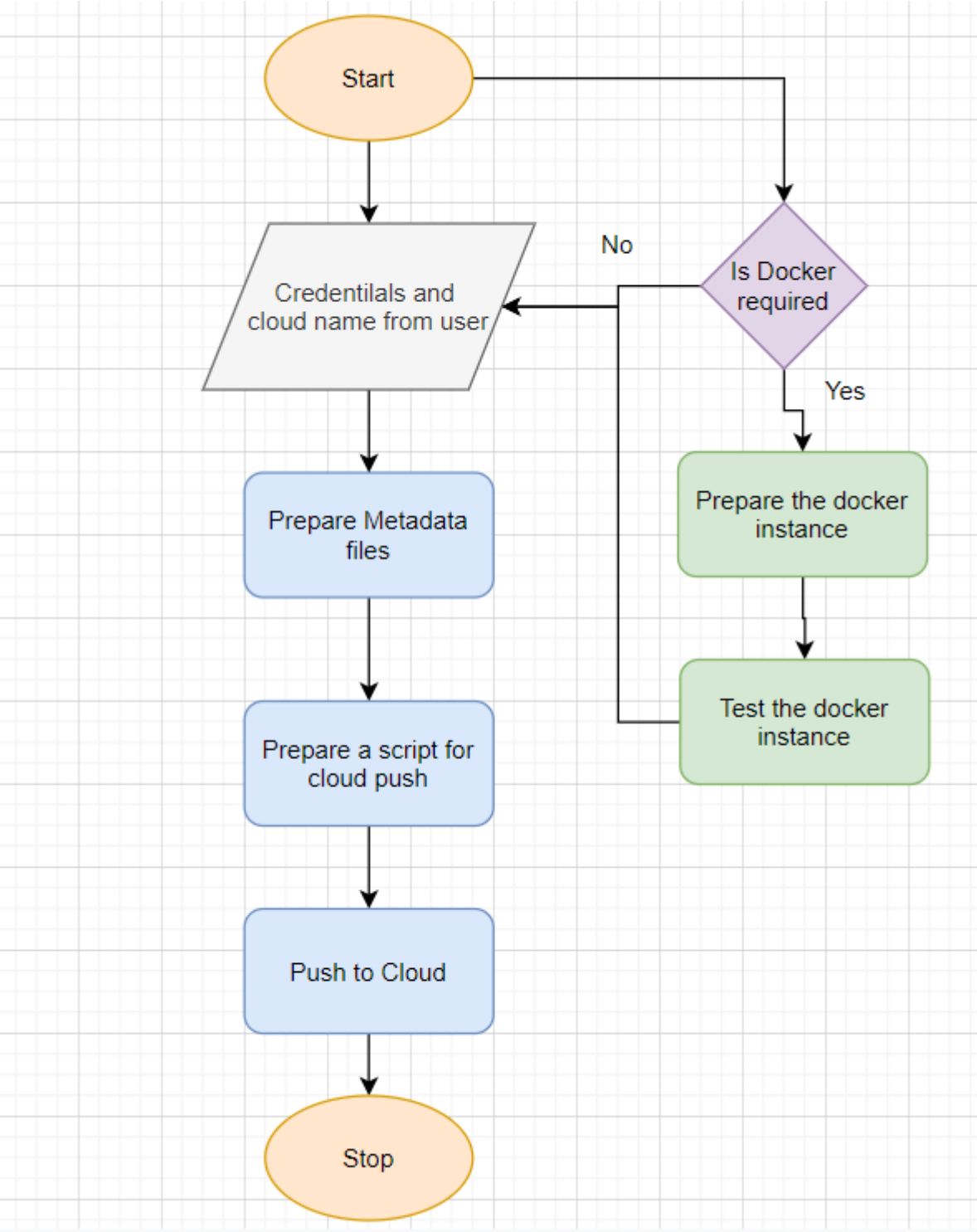


9.2 Exceptions Scenarios Module Wise

Step	Exception	Mitigation
Columns don't match in training and Prediction data	Show error message	The user enters the correct data

10 Deployment Strategy

10.1 Technical solution design



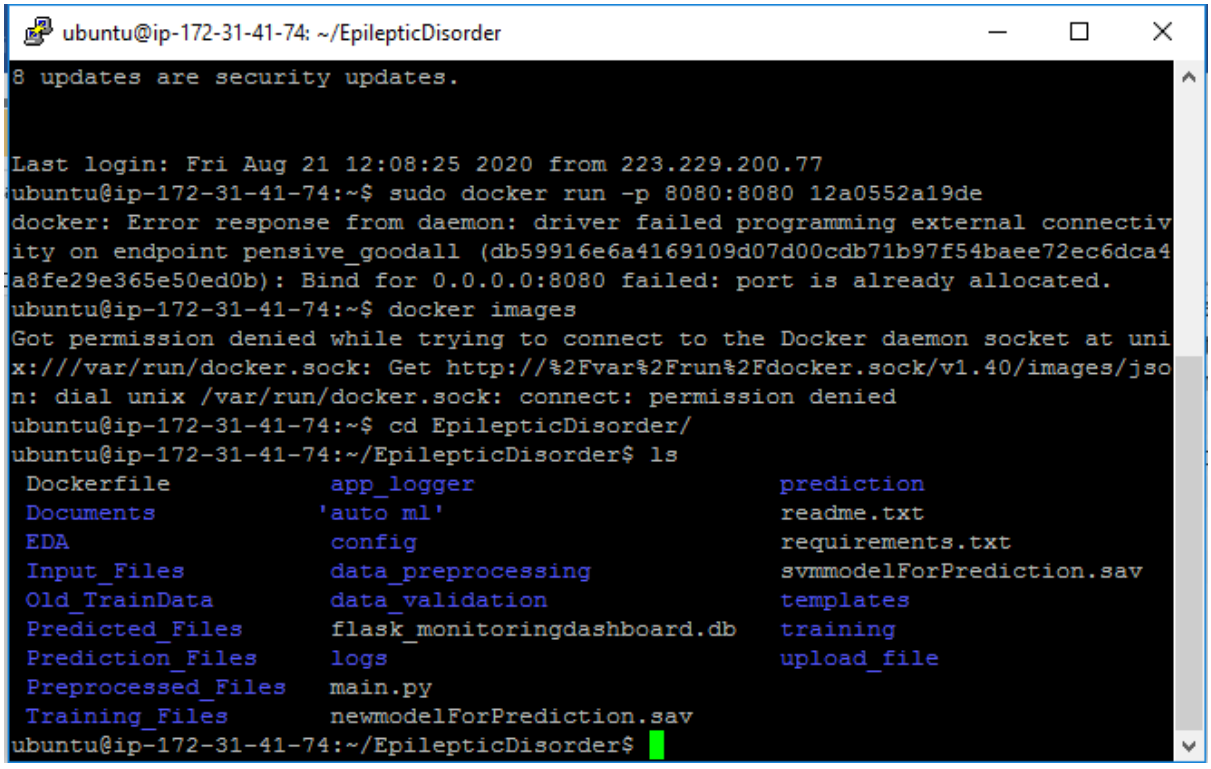
10.2 Exceptions Scenarios Module Wise

Step	Exception	Mitigation
Wrong Cloud credentials	Show error message	The user enters the correct data
Docker instance not working	Show error message	Fix the error
Cloud push failed	Show the error	Make corrections to the metadata files
Cloud app not starting		Ask the user for cloud logs for debugging

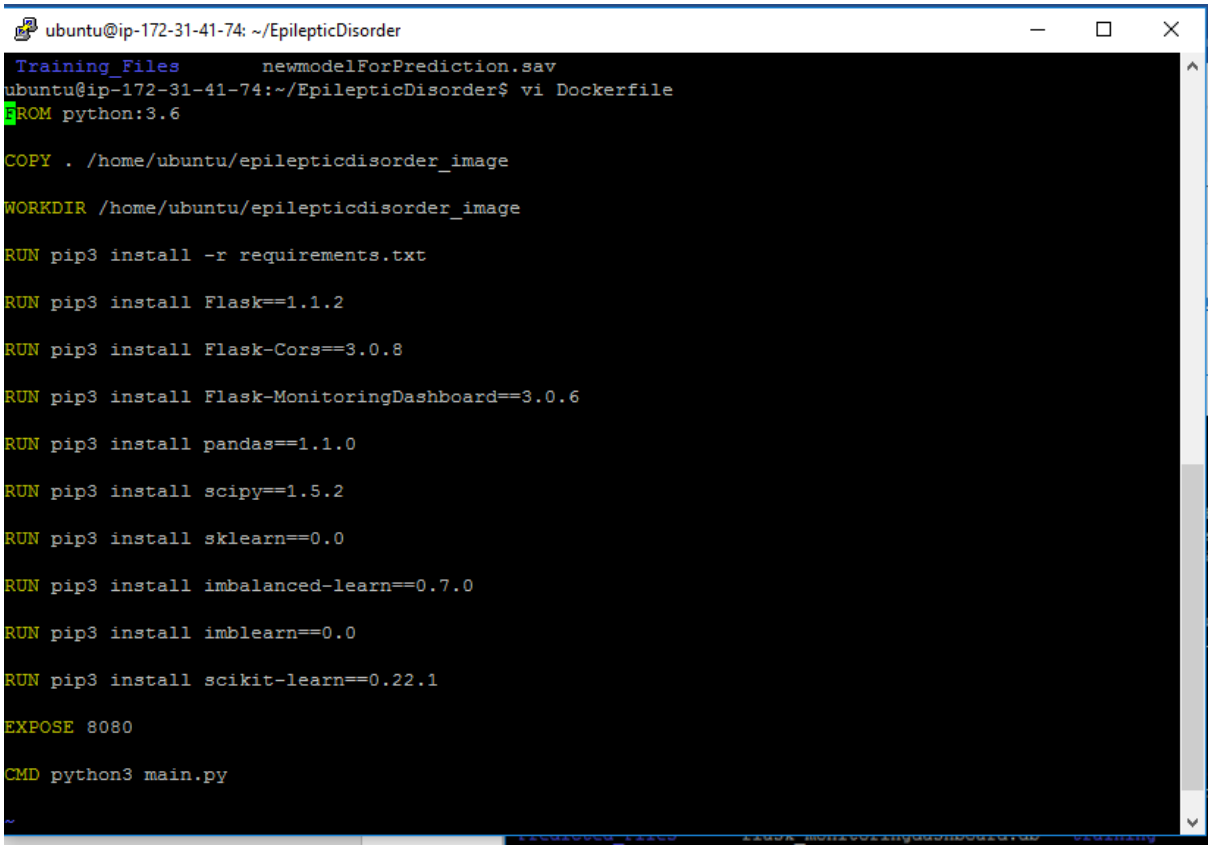
The App has been containerized using Docker in EC2 instance in AWS.

The URL to access the App is http://ec2-13-232-114-74.ap-south-1.compute.amazonaws.com:8080/

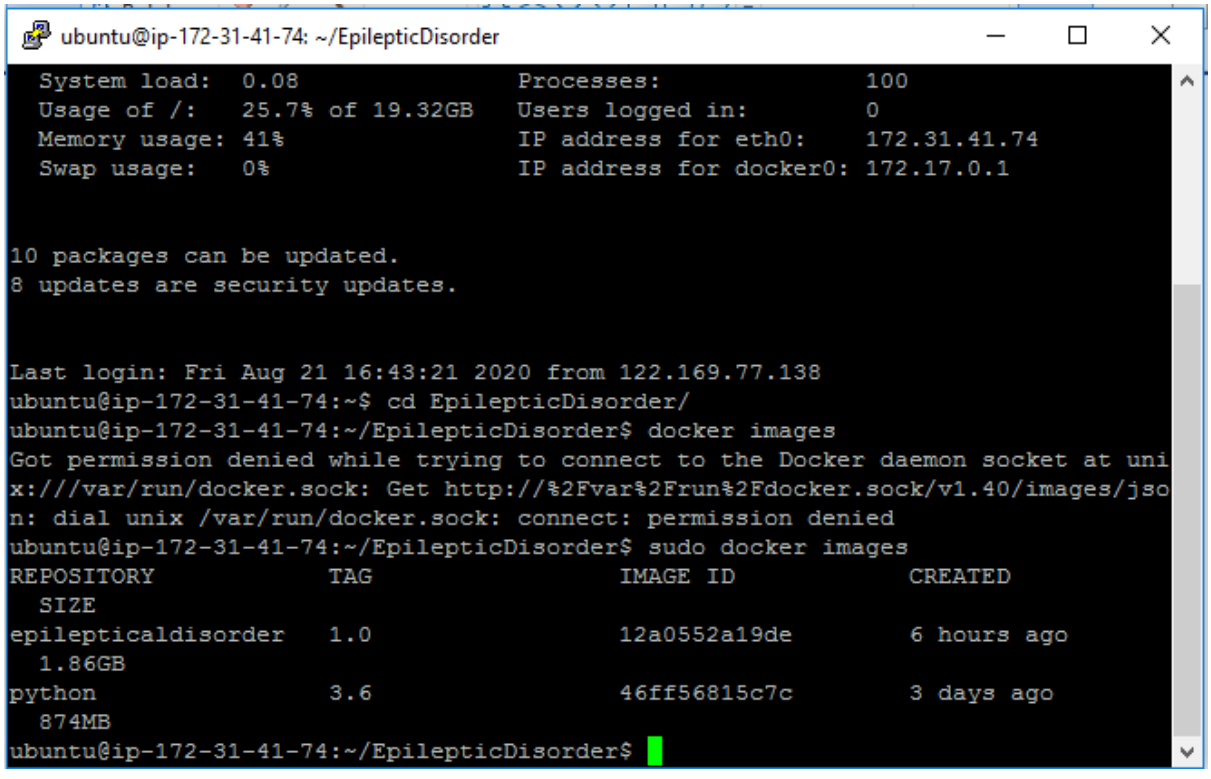
Docker file created in the project.



View of the Docker file



View of the Docker Image for Epileptic Disorder



11 Logging

Logging of every step

Entry to the methods

Exit from the methods with success/ failure message

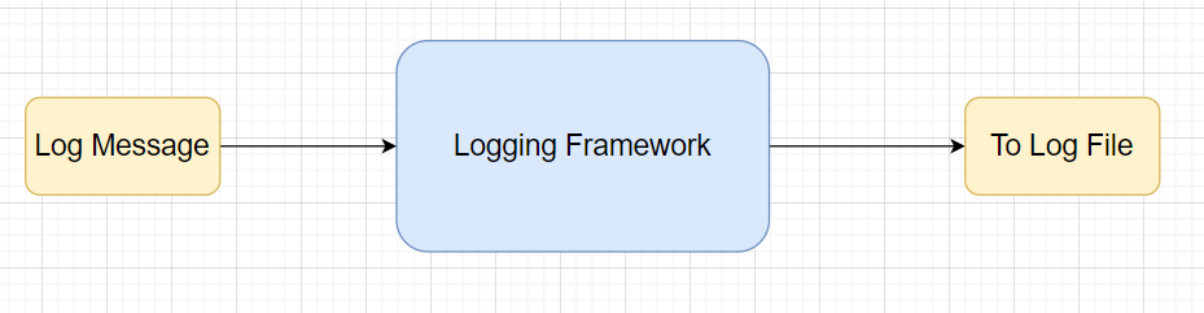
Error message Logging

Training start and end

Prediction start and end

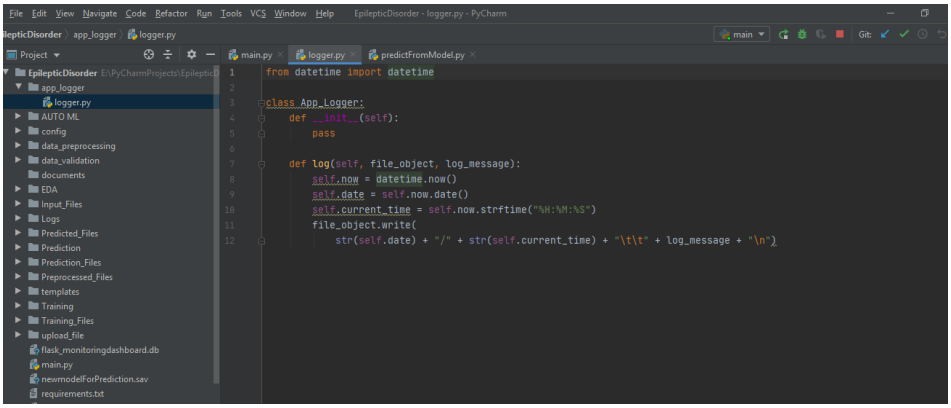
Achieve asynchronous logging

11.1 Technical solution design



11.2 Exceptions Scenarios Module Wise

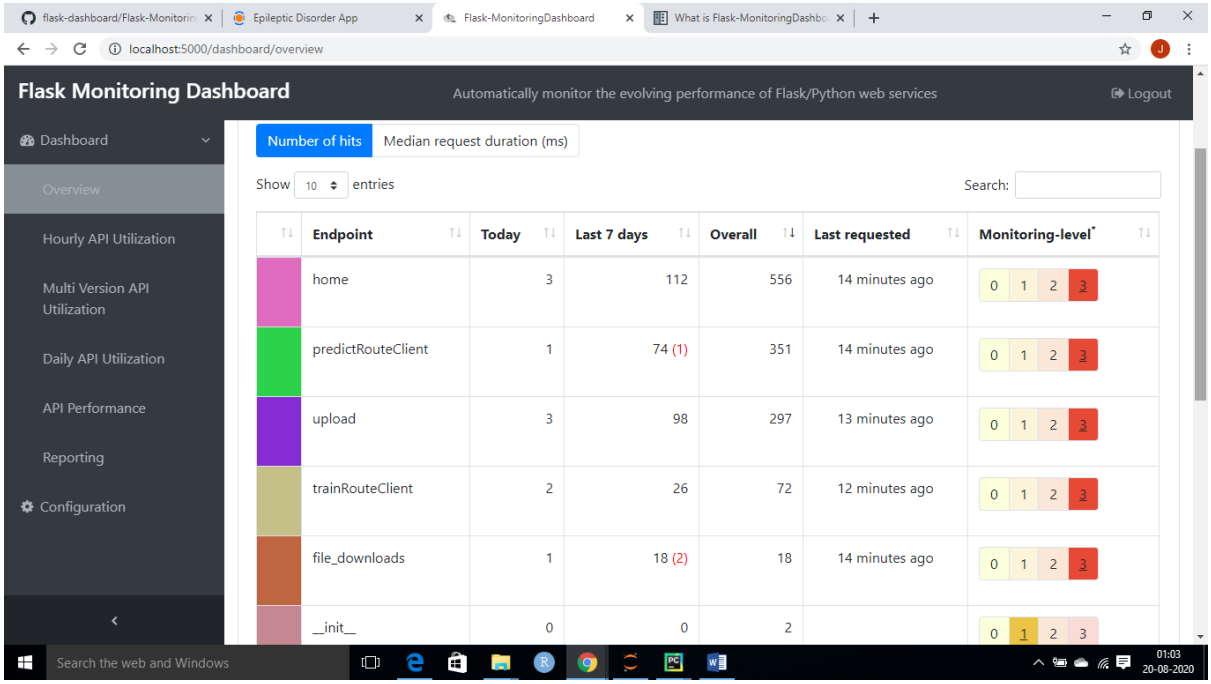
Ideally, the logging should never fail.



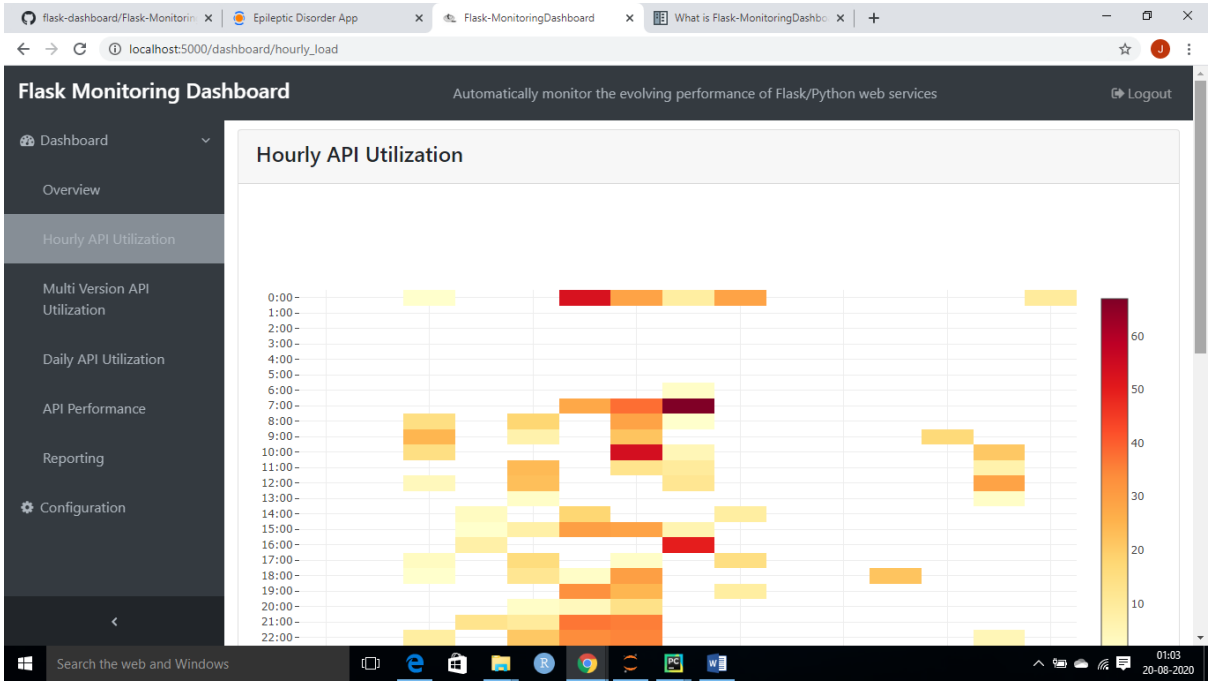
12 Monitoring

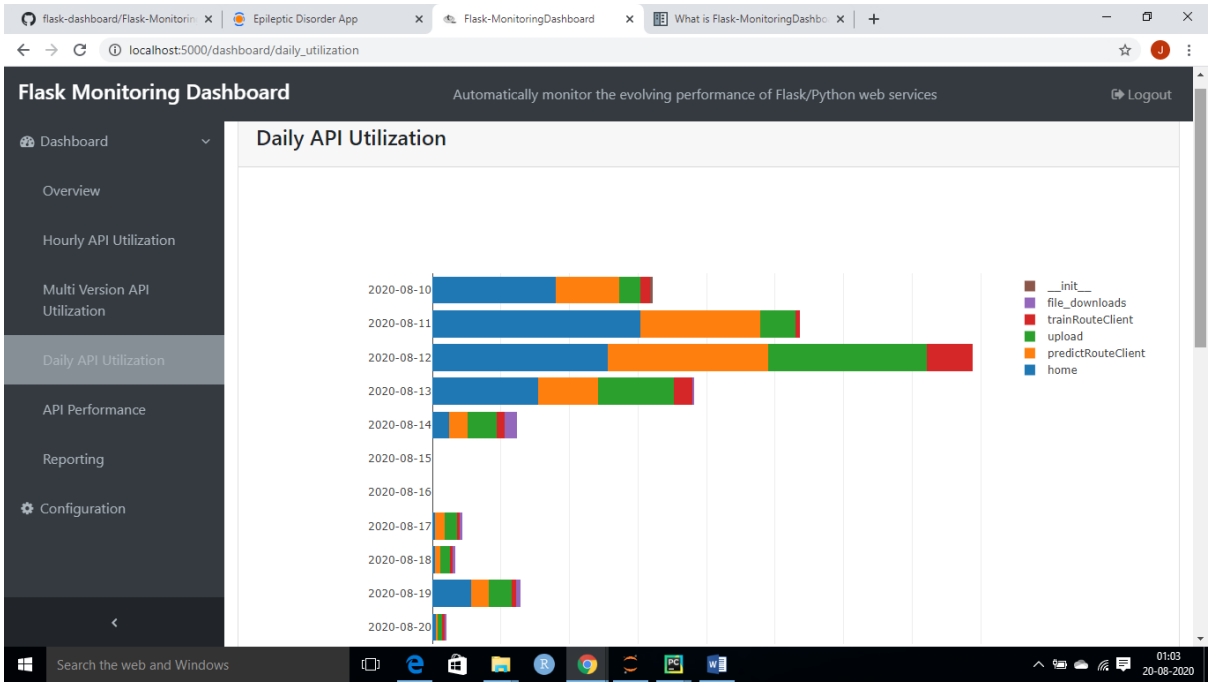
Phase 2

No. of hits for each endpoint

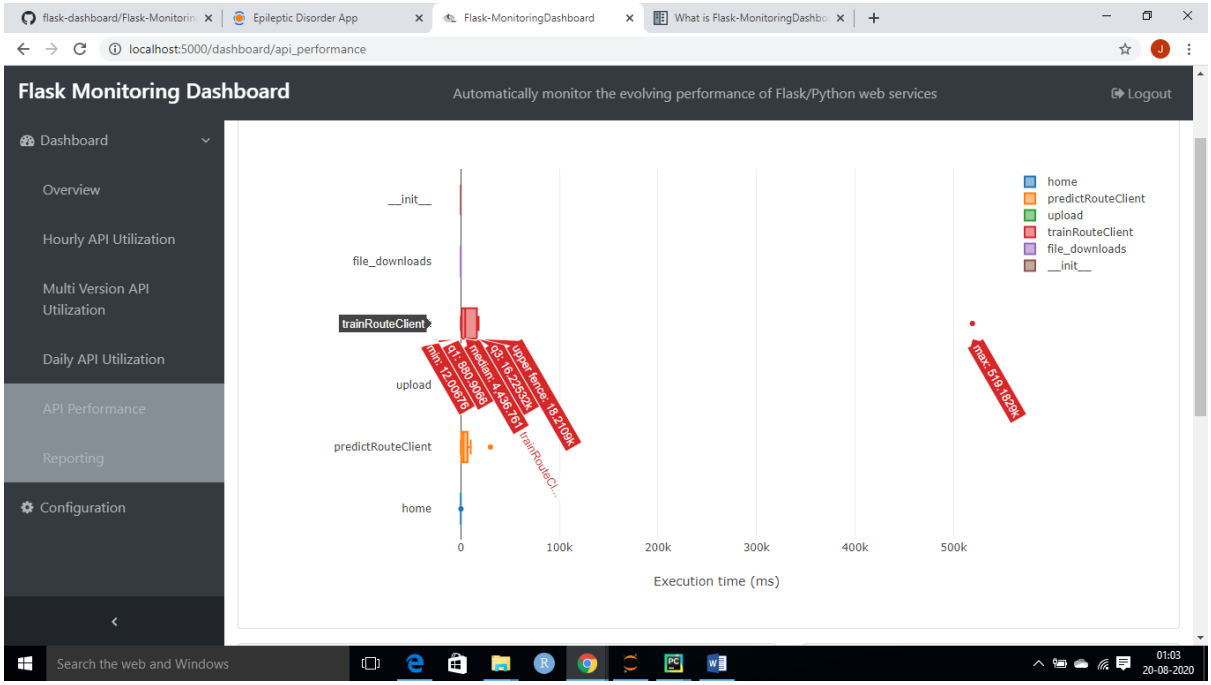


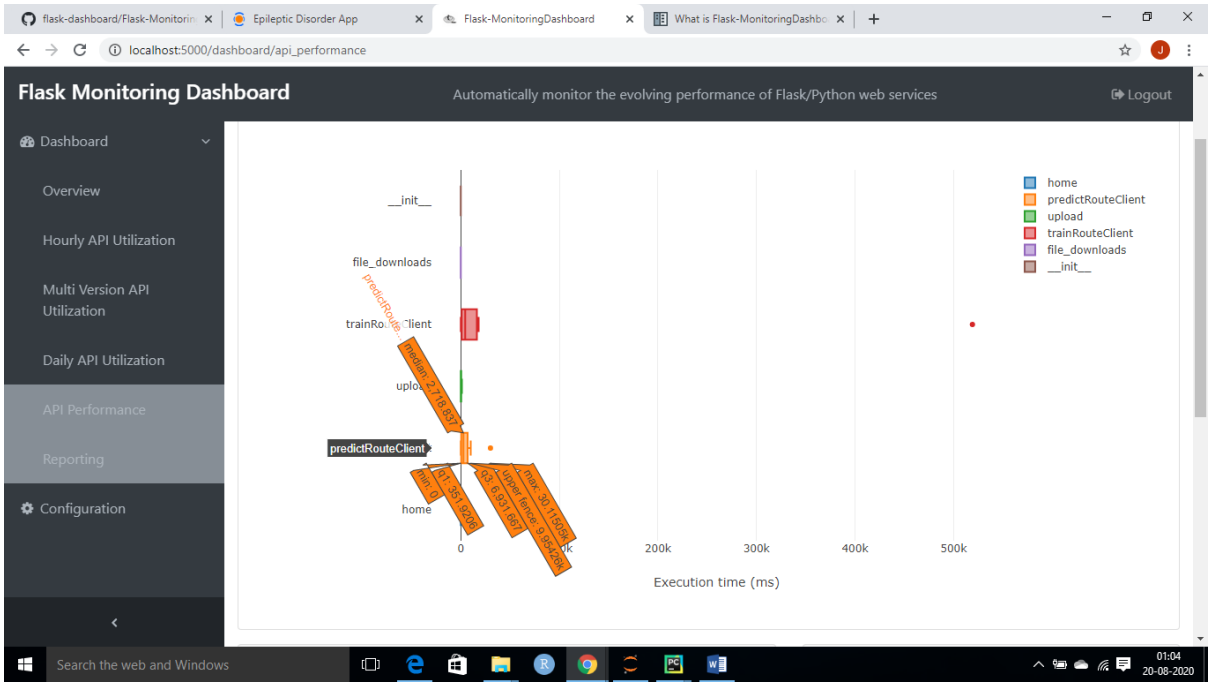
Hourly & Daily API Utilization





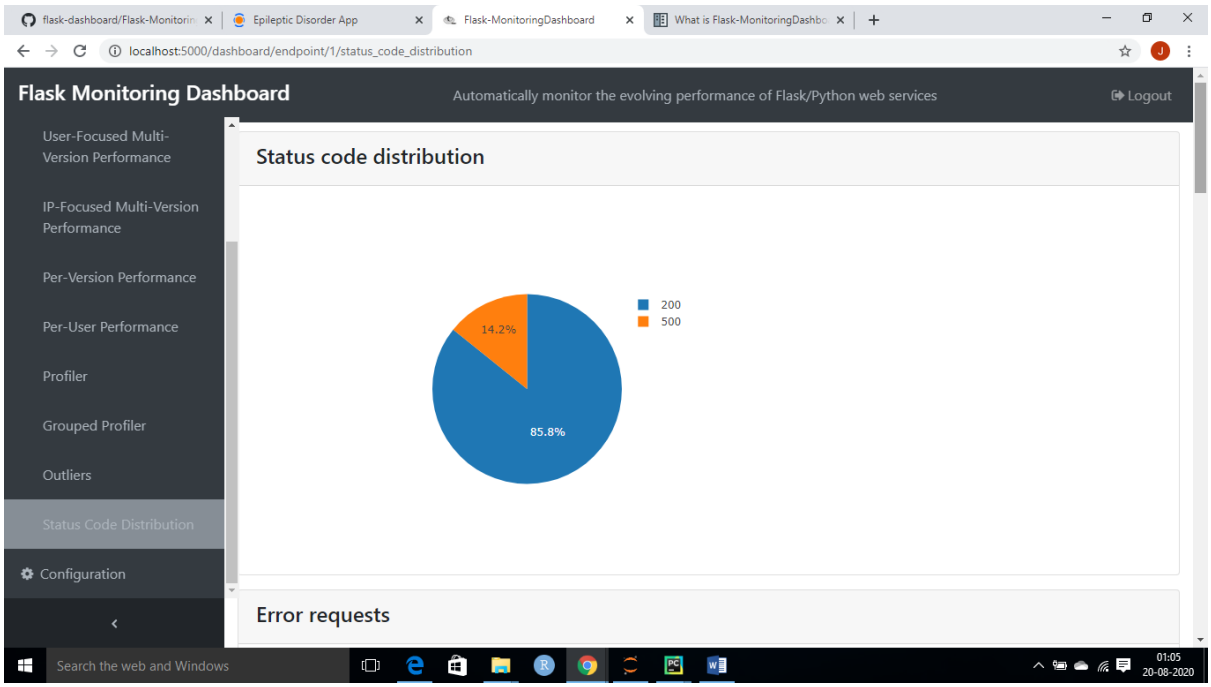
API Performance. Time spent on each route end point



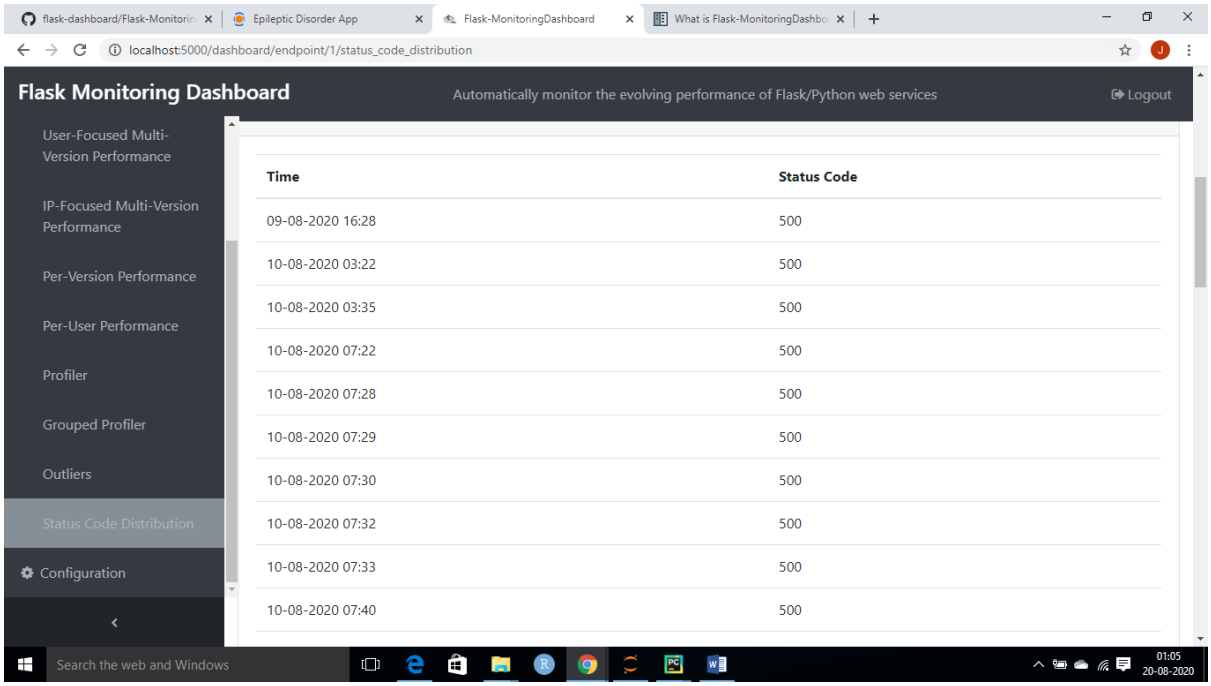


No. of Successes for ‘predictRouteClient’ corresponds to no of predictions

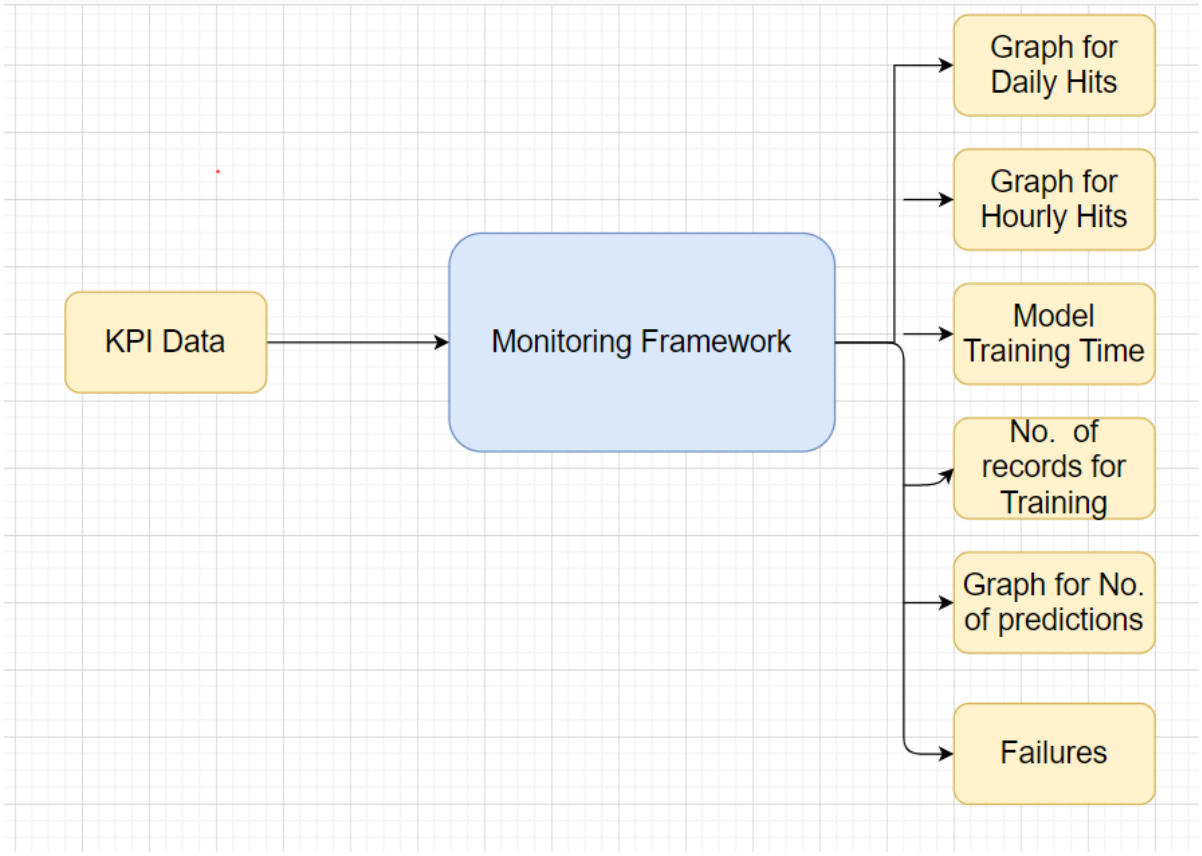
Total no of Failures taken away from Total no. of hits for ‘predictRouteClient’ gives the total no of Predictions. This data can be derived per day, per week etc.



Failures

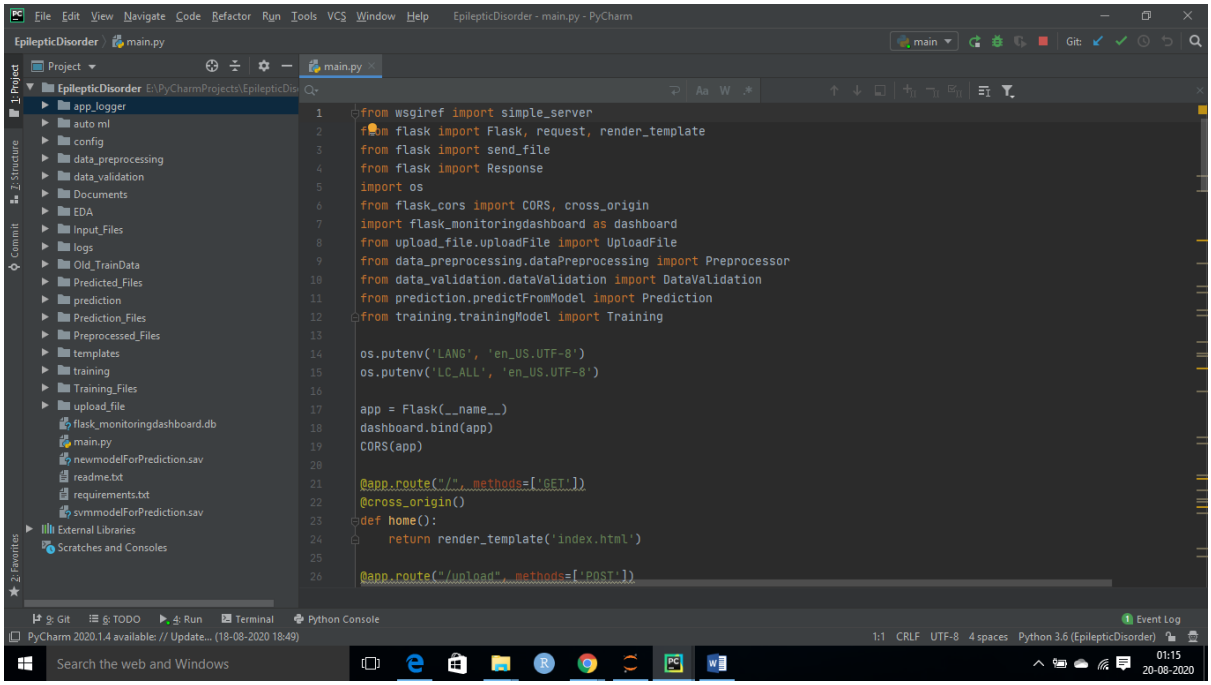


12.1 Technical solution design



12.2 Exceptions Scenarios Module Wise

Ideally, the flast monitoring dashboard should never fail.



13 Hardware Requirements

13.1 Requirements for model training

The minimum configuration should be:

- 8 GB RAM
- 2 GB of Hard Disk Space
- Intel Core i5 Processor

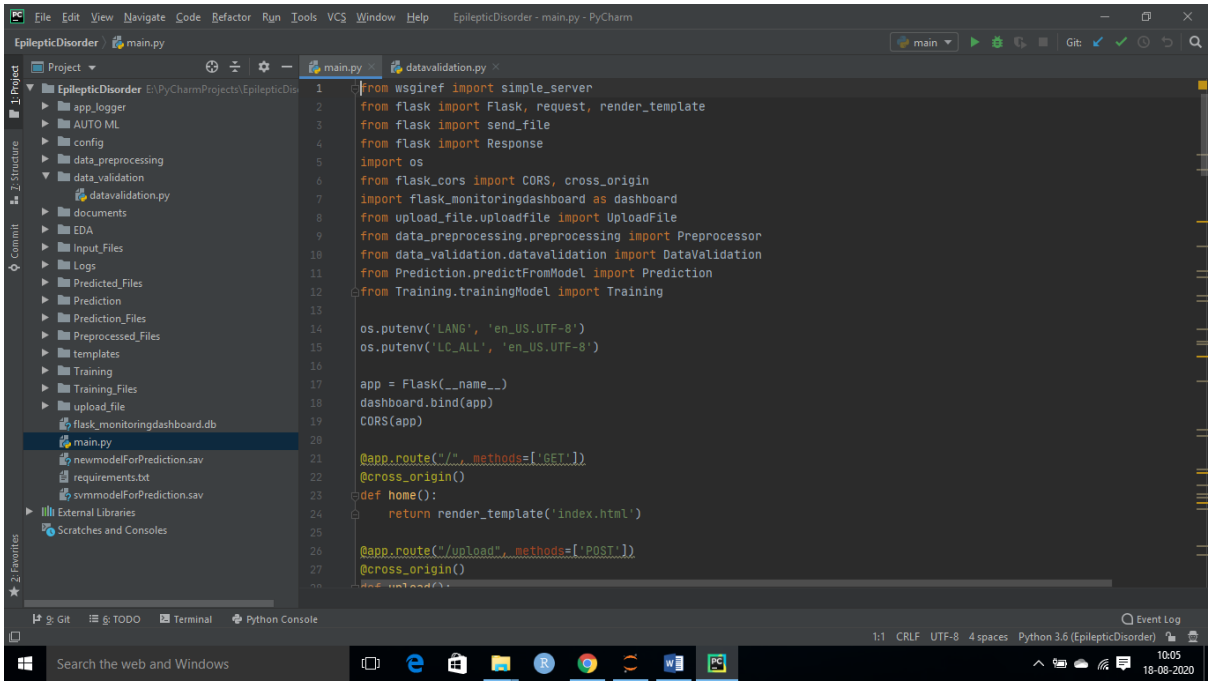
13.2 Requirements for model testing

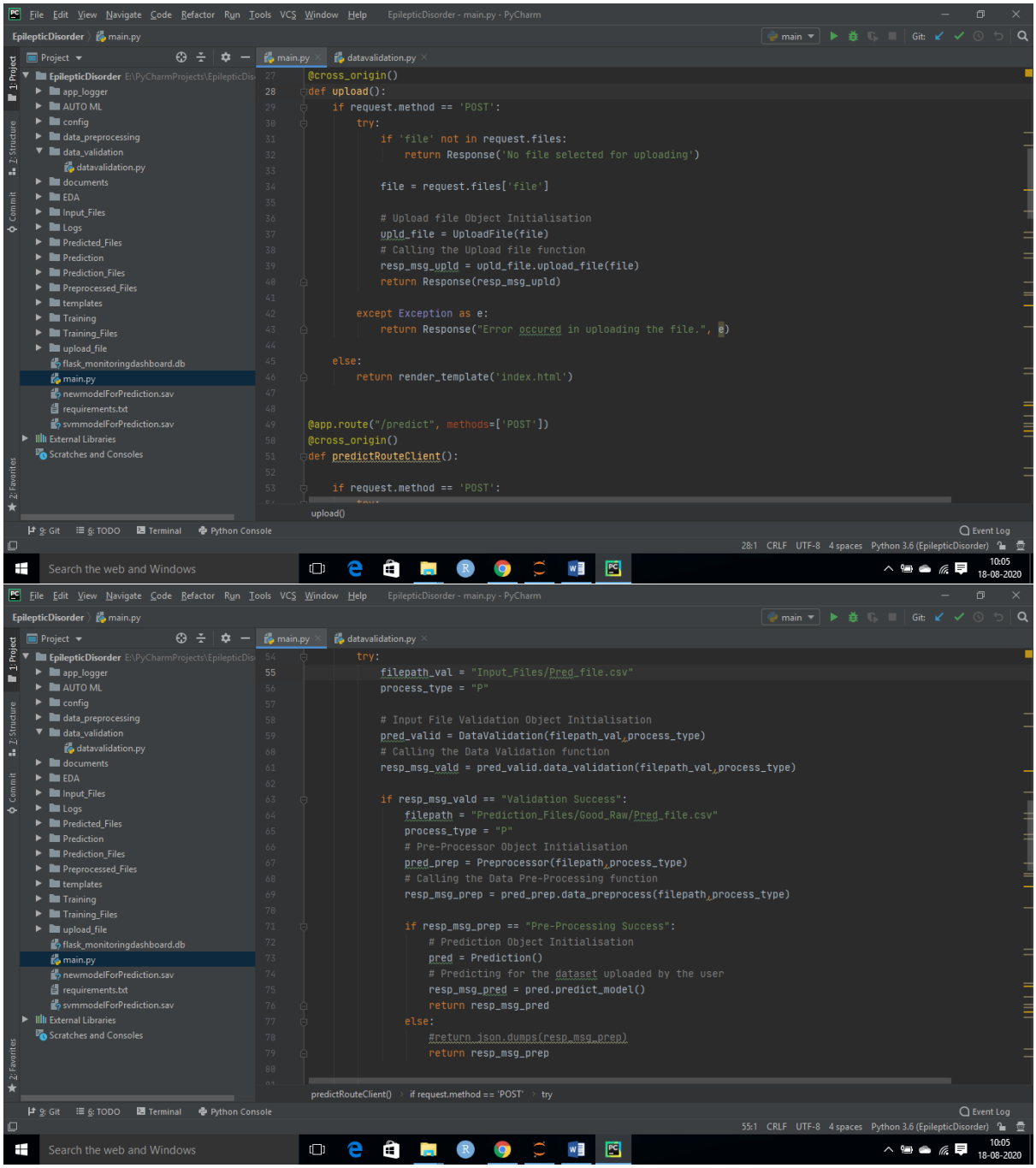
The minimum configuration should be:

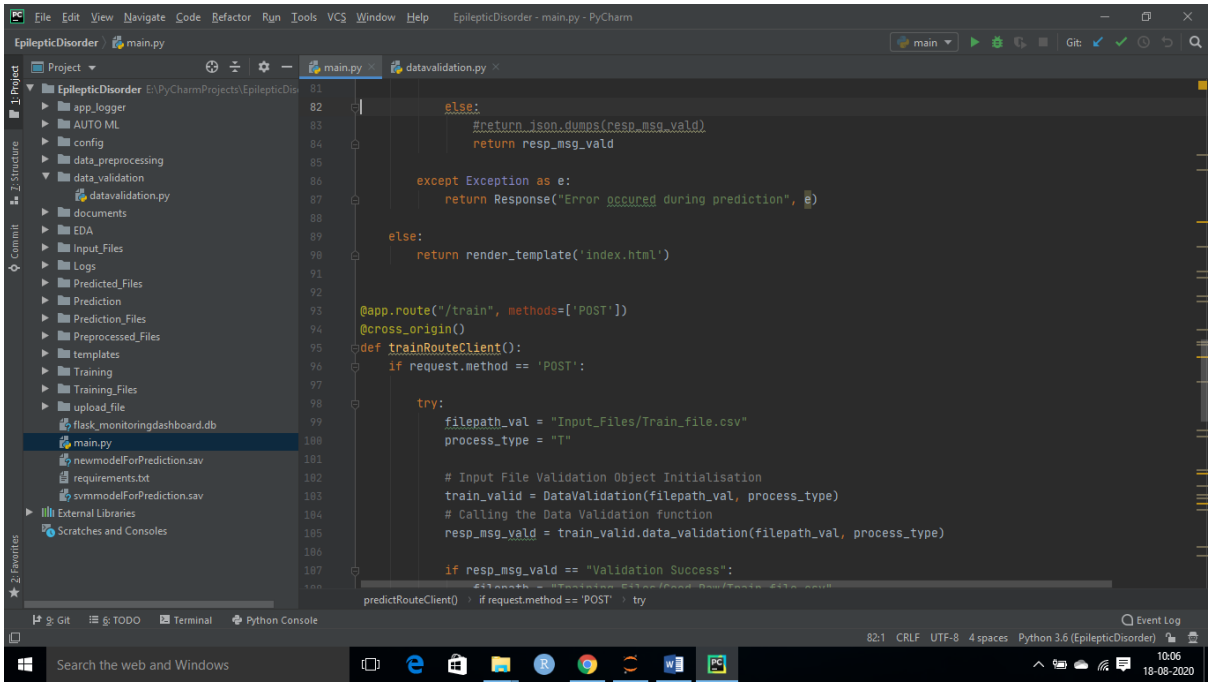
- 4 GB RAM
- 2 GB of Hard Disk Space
- Intel Core i5 Processor

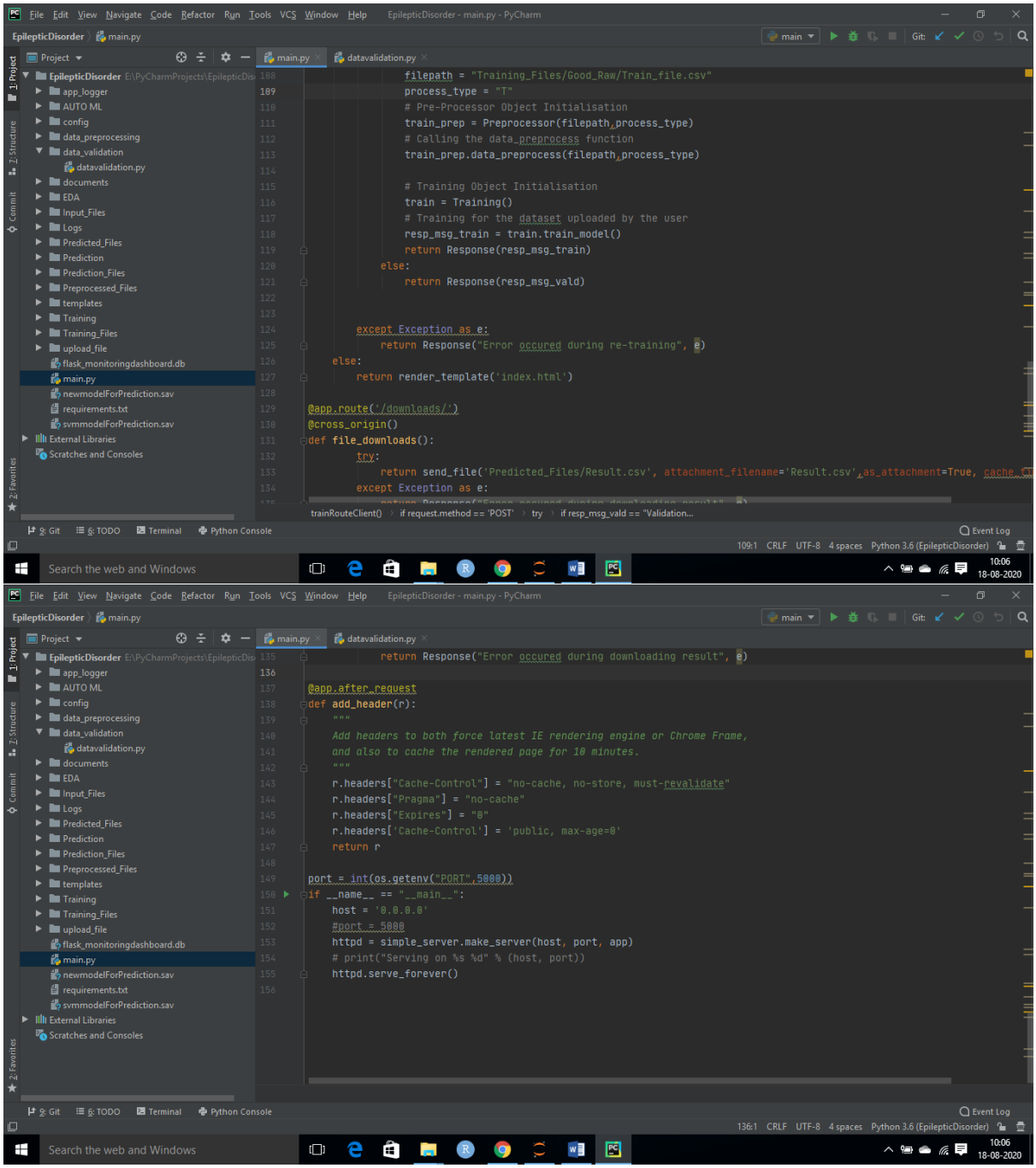
14 Sample code and standard to be followed:

Sample Code:




File Edit View Navigate Code Refactor Run Tools VCS Window Help
EpilepticDisorder - main.py - PyCharm
EpilepticDisorder
main.py
main
Git:
Project
main.py
datavalidation.py
EpilepticDisorder E:\PyCharmProjects\EpilepticDisorder
app_logger
AUTO ML
config
data_preprocessing
data_validation
datavalidation.py
documents
EDA
Input_Files
Logs
Predicted_Files
Prediction
Prediction_Files
Preprocessed_Files
templates
Training
Training_Files
upload_file
flask_monitoringdashboard.db
main.py
newmodelForPrediction.sav
requirements.txt
svmmodelForPrediction.sav
External Libraries
Scratches and Consoles
@cross_origin()
def upload():
if request.method == 'POST':
try:
if 'file' not in request.files:
return Response('No file selected for uploading')
file = request.files['file']
# Upload file Object Initialisation
upld_file = UploadFile(file)
# Calling the Upload file function
resp_msg_upld = upld_file.upload_file(file)
return Response(resp_msg_upld)
except Exception as e:
return Response("Error occured in uploading the file.", e)
else:
return render_template('index.html')
@app.route("/predict", methods=['POST'])
@cross_origin()
def predictRouteClient():
if request.method == 'POST':
try:
upload()
Git
TODO
Terminal
Python Console
Event Log
28:1 CRLF UTF-8 4 spaces Python 3.6 (EpilepticDisorder)
Search the web and Windows
10:05
18-08-2020
File Edit View Navigate Code Refactor Run Tools VCS Window Help
EpilepticDisorder - main.py - PyCharm
EpilepticDisorder
main.py
main
Git:
Project
main.py
datavalidation.py
EpilepticDisorder E:\PyCharmProjects\EpilepticDisorder
app_logger
AUTO ML
config
data_preprocessing
data_validation
datavalidation.py
documents
EDA
Input_Files
Logs
Predicted_Files
Prediction
Prediction_Files
Preprocessed_Files
templates
Training
Training_Files
upload_file
flask_monitoringdashboard.db
main.py
newmodelForPrediction.sav
requirements.txt
svmmodelForPrediction.sav
External Libraries
Scratches and Consoles
try:
filepath_val = "Input_Files/Pred_file.csv"
process_type = "P"
# Input File Validation Object Initialisation
pred_valid = DataValidation(filepath_val,process_type)
# Calling the Data Validation function
resp_msg_vald = pred_valid.data_validation(filepath_val,process_type)
if resp_msg_vald == "Validation Success":
filepath = "Prediction_Files/Good_Raw/Pred_file.csv"
process_type = "P"
# Pre-Processor Object Initialisation
pred_prep = Preprocessor(filepath,process_type)
# Calling the Data Pre-Processing function
resp_msg_prep = pred_prep.data_preprocess(filepath,process_type)
if resp_msg_prep == "Pre-Processing Success":
# Prediction Object Initialisation
pred = Prediction()
# Predicting for the dataset uploaded by the user
resp_msg_pred = pred.predict_model()
return resp_msg_pred
else:
#return json.dumps(resp_msg_prep)
return resp_msg_prep
predictRouteClient() if request.method == 'POST' try
Git
TODO
Terminal
Python Console
Event Log
55:1 CRLF UTF-8 4 spaces Python 3.6 (EpilepticDisorder)
Search the web and Windows
10:05
18-08-2020


File Edit View Navigate Code Refactor Run Tools VCS Window Help
EpilepticDisorder - main.py - PyCharm
EpilepticDisorder
main.py
main
Git:
Project
main.py
datavalidation.py
EpilepticDisorder E:\PyCharmProjects\EpilepticDis
app_logger
AUTO ML
config
data_preprocessing
data_validation
datavalidation.py
documents
EDA
Input_Files
Logs
Predicted_Files
Prediction
Prediction_Files
Preprocessed_Files
templates
Training
Training_Files
upload_file
flask_monitoringdashboard.db
main.py
newmodelForPrediction.sav
requirements.txt
svmmodelForPrediction.sav
External Libraries
Scratches and Consoles
else:
#return json.dumps(resp_msg_vald)
return resp_msg_vald
except Exception as e:
return Response("Error occured during prediction", e)
else:
return render_template('index.html')
@app.route("/train", methods=['POST'])
@cross_origin()
def trainRouteClient():
if request.method == 'POST':
try:
filepath_val = "Input_Files/Train_file.csv"
process_type = "T"
# Input File Validation Object Initialisation
train_valid = DataValidation(filepath_val, process_type)
# Calling the Data Validation function
resp_msg_vald = train_valid.data_validation(filepath_val, process_type)
if resp_msg_vald == "Validation Success":
predictRouteClient() if request.method == 'POST' try
Git TODO Terminal Python Console
Event Log
82:1 CRLF UTF-8 4 spaces Python 3.6 (EpilepticDisorder)
Search the web and Windows
10:06
18-08-2020



Coding Standard:

1. Imports should usually be on separate lines
2. Avoid trailing whitespace anywhere. Because it's usually invisible, it can be confusing.
3. Compound statements (multiple statements on the same line) are generally discouraged
4. Comments should be complete sentences. Always make a priority of keeping the comments up-to-date when the code changes. Ensure that your comments are clear and easily understandable to other speakers of the language you are writing in.

5. Never use the characters 'l' (lowercase letter el), 'O' (uppercase letter oh), or 'I' (uppercase letter eye) as single character variable names.
6. The name of the variables should start with small case capital letters and a multi word variable should be named as: word1_word2_word3.
7. The variable name should be appropriate based on the things that they do. DO NOT USE NAMES LIKE x, k, y etc. Always use a meaningful English word. For example, customer_name, nearest_neighbour etc.
8. Method names should start with small case characters. They should start with a verb and make a meaningful sense of what they are supposed to accomplish. For e.g.: load_data_from_sql()
9. Always use `self` for the first argument to instance methods.
10. Class names should normally use the CapWords convention. Class name should also represent the functionality of the class. For e.g. DataLoader()
11. Modules/Packages/Folders should have short, all-lowercase names. Underscores can be used in the module name if it improves readability. For e.g.: data_ingestion
12. Constants are usually defined on a module level and written in all capital letters with underscores separating words. Examples include `MAX_OVERFLOW` and `TOTAL`.
13. Comparisons to singletons like None should always be done with `is` or `is not`, never the equality operators
14. The code should be properly enclosed withing try and exception blocks and the exceptions should be handled with proper error messages.
15. Additionally, for all try/except clauses, limit the `try` clause to the absolute minimum amount of code necessary. Again, this avoids masking bugs
16. When a resource is local to a particular section of code, use a `with` statement to ensure it is cleaned up promptly and reliably after use.
17. Be consistent in return statements. Either all return statements in a function should return an expression, or none of them should. If any return statement returns an expression, any return statements where no value is returned should explicitly state this as `return None`, and an explicit return statement should be present at the end of the function (if reachable)
18. Object type comparisons should always use isinstance() instead of comparing types directly
19. Don't compare boolean values to True or False using `==`