RAPIDML

A PROJECT REPORT
(15CS752 – Software Application Development Lab)

Submitted by

Kalathiappan K(201604042)
Visanth VK(201604119)
Vivek Raja PS(201604121)

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

MEPCO SCHLENK ENGINEERING COLLEGE, SIVAKASI
(AUTONOMOUS)

ANNA UNIVERSITY: CHENNAI 600 025

OCTOBER 2019

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **“RAPIDML”** is the bonafide work of “KALATHIAPPAN.K(201604042),VISANTH.V.K(201604119),VIVEKRAJA.P.S (201604121)**”** who carried out the project work in 15CS752 – Software Application Development Lab.

SIGNATURE	SIGNATURE
Dr. K.MUNEESWARAN, M.E., Ph.D.	**MR. B.SELVAKUMAR, M.E., (Ph.D.)**
Sr. Prof. and Head of the Department Dept. of Computer Science and Engg. Mepco Schlenk Engg. College (Autonomous), Sivakasi, Virudhunagar Dt.-626 005	Faculty Incharge AP/CSE Dept. of Computer Science and Engg. Mepco Schlenk Engg. College (Autonomous), Sivakasi, Virudhunagar Dt.-626 005

Submitted to Viva-Voce examination held on .... /...... / **2019**

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ABSTRACT

Nowadays, large amount of data is available everywhere. Therefore, it is very important to analyze this data in order to extract some useful information and to develop an algorithm based on this analysis. This can be achieved through machine learning. Machine learning is an integral part of artificial intelligence, which is used to design algorithms based on the data trends and historical relationships between data. Machine learning is used in various fields such as bioinformatics, intrusion detection, Information retrieval, game playing, marketing, malware detection, image deconvolution and so on.

This system provide UI tool for developing machine learning without involving in programming. The Web application is mainly designed for the purpose to be used by Mathematicians, Data Scientists and other non-programmers and they can train and export their Machine Learning models without a single line of coding. The entire process of preprocessing, model training, visualization, hyper parameter tuning and model export can be done using the web application without doing a single line of code.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 PRODUCT PERSPECTIVE

The Application for Machine learning is a Web based Application Created on Django Platform Which enable user to develop machine learning model. The scope of this application is for the person who want to develop machine learning model in faster manner.

The user just upload the csv file and develop

1.2 PRODUCT FEATURES

- The Application provide good interface so that user can easily perform there operations like preprocessing and train a model.
- User can visualize the data set what they given.
- Users have several options in model development phase like SVM and so on.
- And they can test the accuracy for the developed model in testing phase.
- And hyper parameter tuning for the model developed also included in this system so user need not to worry about coding part.

CHAPTER 2

REQUIREMENTS DESCRIPTION

2.1 FUNCTIONAL REQUIREMENTS

Functional Requirements are the core requirements that a product possesses. **Table 2.1** shows all the functional requirements of the system

ID	Functional Requirements
1	**Preprocessing:** User can upload the data set and they can easily preprocess it. Like they can replace the NAN,? And some value with mean, median or custom input. And they can intimated that where the class is categorical or not
2	**Visualization:** The visualization part is also two types. i.e. the data visualization before model building on which the mean, median, mode, max, min and outliers of each attribute are displayed. And then after model training, the confusion matrix, ROC curve and ground truth and predicted values and plotted in a graph and displayed.
3	**Model Training:** Based on the class labels whether they are categorical or discrete, the respective models are only displayed. The regression or classification models are displayed as per the type of data set uploaded and then they are trained with respect to default set of parameters.
4	**Hyper Parameter Tuning:** The models are in default trained using the default parameters. But in order to increase the model's performance, the parameters can be adjusted. The portal provides facilities to gives parameter and trains the models on that parameters and hyper-parameter tuning on a set of parameters can also be done.

Table 2.1 Functional requiremnts of the system.

2.2 NON FUNCTIONAL REQUIREMENTS

Non-Functional requirements are those which are not part of the core features but they result in the final product reach. Some of them are Security, Reliability etc. **Table 2.2** shows the non functional requirements of the system

ID	**Non functional requirements**	**Description**
1	**Interoperability**	This application is a web-based application and it operate in any kind of PCs/Laptops/Mobile Phones irrespective of manufacture of devices.
2	**Reliability**	While processing the user request,if any error occurs this application is able to show notification.
3	**Availability**	This Application is available at any time and provide service if a user access it.

Table 2.2 Non-Functional requirements

2.3 INTERFACE REQUIREMENTS

The **Table 2.3** shows the interfaces needed to communicate with our app. As this is an web application, the user can communicate through an PC/Laptop/mobile. The users also need to connect to the internet as the details are retrieved from the centralized database.

Id	**Interface**	**Requirements**
IR1	Web	This web application is compatible with the entire PCs/Laptops/Mobile Phones

Table 2.3: Interface Requirements

CHAPTER 3
SYSTEM DESIGN

3.1 USER INTERFACE DESIGN

3.1.1 Home screen:

This Screen Show the landing page of RapidML and user can upload the dataset in that page. The dataset has to be in csv format. Other data formats files will throw up an exception.

3.1.2 Preprocessing Page:

This Screen shows what are the features and class in the dataset and we will then get into preprocessing phase on the screen. There are several parts in preprocessing. They are replacing the missing values. The missing values can be anything. They can be NaN, ? or any other value that changes as per any dataset. So an additional feature of adding the custom missing value type is also given. And replace with feature in which we have to select what value should replace the missing value specified in the missing value column. The default available replacing values are mean, median and mode, but you can also provide custom values to replace the missing value by specifying the custom value in a text box given for custom value.

The next feature available is drop columns which can be used to drop some unwanted columns that are not relevant to the scenario. The specific column or attributes can be dropped by just clicking the radio button along the respective feature. The last facility available is categorical feature or not. If a particular feature is categorical and if it's needed to be label encoded, them the particular column's categorical radio button should be checked so that then label encoding is applied onto it and then it can be proceeded to model training process.

3.1.3 Train Page:

After preprocessing stage, training stage, where based on the dataset uploaded it is identified whether it's a regression or classification task and only those models are displayed. The regression models available are Linear Regression, Lasso Regression and Ridge Regression etc. The classification models included in the site are SVM, Decision Tree, Logistic Regression, Naïve Bayesian and knn. Then the split ratio has to be selected. After clicking on the train button, the model will be trained on the default set of parameters and the given split ratio and the resulting metric scores are returned.

3.1.4 Visualization Page:

The visualization part is also two types. i.e. the data visualization before model building on which the mean, median, mode, max, min and outliers of each attribute are displayed.

And then after model training, the confusion matrix, ROC curve and ground truth and predicted values and plotted in a graph and displayed.

The mean. median, mode, min and max are done using the description method from pandas library which is used to read the csv data sets. The outliers are also shown using the box plots. The box plots are drawn and then displayed in the visualization page.

After the model training phase, the confusion matrix is displayed to display the TP, TN, FP and FN values. The other visualization features are ROC curve and a plot between ground truth and predicted values.

3.1.5 Hyper parameter tuning Page:

The models are in default trained using the default parameters. But in order to increase the model's performance, the parameters can be adjusted. The portal provides facilities to gives parameter and trains the models on that parameters and hyper-parameter tuning on a set of parameters can also be done.

The parameter specific to a particular model can be given and then trained based on that or a list of parameters can also be given for any parameter and then GridSearch library is used and the best set of parameters that will result the best metrics will be returned.

3.1.6 Testing and model export page:

The testing page is where you can test the model by uploading more test data in csv. This part is crucial to check whether the model will perform in real life environment or not. This will help to prevent over fitting or under fitting of model. After model testing the model metrics are displayed below that.

The other feature is to download the model file. The model object can be converted into string and then dumped into a file using pickle library. Then the trained model can be used i.e. the downloaded pickle file model can be imported anywhere and can be deployed later for real time purposes.

3.2 ARCHITECTURAL DESIGN

Figure 3.1 shows the architectural design of the web application for online EB management system. The user interacts with the web application. The response for the application is received form the server in two forms. Direct interaction with the server, JSON response by invoking the web service from server. The images of products are also retrieved from the server to display them in the application.

When the web service is invoked by the application, the required data is sent to the server by GET method and processed there. After processing the data in server JSON response with either success or failure message is sent to the application.

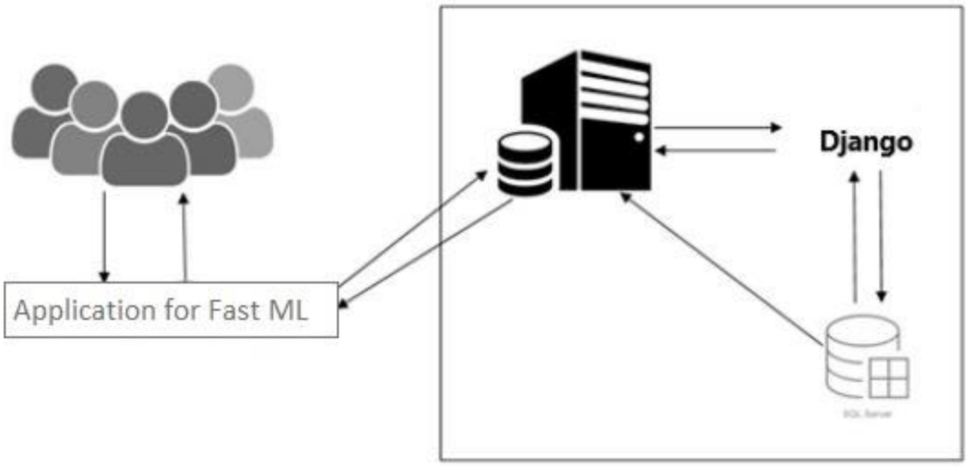


Figure 3.1 Architectural design

3.2.1 Decomposition Description

Table 3.1 Decomposition description for Fileupload

Name:	Dataset Upload
Attributes:	CSV file
Description:	It is used to develop model
Operations:	Fileupload()
Flow of Events:	The File is used and referred throughout the application in all other process.

3

Table 3.2 Decomposition description for Pre-processing

Name:	Preprocessing
Attributes:	CSV file
Description:	It is used to remove unwanted data in the file and make the feature easy to access.
Operations:	Preprocess()
Flow of Events:	The File is used and referred throughout the application in all other process.

Table 3.3 Decomposition description for model Training

Name:	Model Development
Attributes:	Preprocessed Data
Description:	It is used to Develop the model.
Operations:	Develop()
Flow of Events:	The File is used and referred throughout the application in all other process.

Table 3.4 Decomposition description for hyper parameter tuning

Name:	Hyper Parameter Tuning
Attributes:	Model
Description:	Avoid over fit and increase accuracy
Operations:	Tune()
Flow of Events:	The Data is used and referred throughout the application in all other process.

Table 3.5 Decomposition description for Visualization

Name:	Visualization
Attributes:	Dataset
Description:	Visualize the dataset by the image generated by matplotlib module in python
Operations:	Visualize()
Flow of Events:	.

3.3 LOW LEVEL DESIGN

3.3.1 Function specification

Table 3.6 shows a detailed description about the functionUploadFile which is done when user wants to upload the dataset. From this page, the user may redirect to preprocessing page for preprocess the dataset.

Table 3.6 Upload file

Name:	Upload file
File Name:	views.py
Naming Convention:	-
Short Description:	This is used to upload the dataset from your local machine to the Rapid ML system
Container Component:	-
Arguments:	File.
Return:	-
Pre-Condition:	The File should be in .csv format
Post-Condition:	-
Exception:	Exception throws when we not choose the file.

Table 3.7 Preprocessing

Name:	Pre-processing
File Name:	preprocess.py
Naming Convention:	-
Short Description:	This page is for pre-processing the features given in the dataset.
Container Component:	-
Arguments:	Dataset
Return:	
Pre-Condition:	The user should upload the csv file.
Post-Condition:	The dataset will be in the pre-processed state.
Exception:	-

Table 3.7 shows a detailed description about the preprocessing. There are several parts in preprocessing. They are replacing the missing values. The missing values can be anything. They can be NaN, ? or any other value that changes as per any dataset. So an additional feature of adding the custom missing value type is also given. And replace with feature in which we have to select what value should replace the missing value specified in the missing value column.

Table 3.8 Train Model

Name:	Train Model
File Name:	Train.py
Naming Convention:	-
Short Description:	This page is for Train the model to the given dataset
Container Component:	-
Arguments:	Dataset , Model name
Return:	
Pre-Condition:	The date set should be in pre-processed state
Post-Condition:	The test tcore will be updated and model will be trained
Exception:	-

Table 3.8 after preprocessing stage, training stage, where based on the dataset uploaded it is identified whether it’s a regression or classification task and only those models are displayed. The regression models available are Linear Regression, Lasso Regression and Ridge Regression etc. The classification models included in the site are SVM, Decision Tree, Logistic Regression, Naïve Bayesian and knn. Then the split ratio has to be selected. After clicking on the train button, the model will be trained on the default set of parameters and the given split ratio and the resulting metric scores are returned.

Table 3.9 Hyper Parameter Tuning

Name:	Hyper Parameter Tuning
File Name:	Tune
Naming Convention:	-
Short Description:	This page is for tune the default parameter. To increase the performance of the algorithm.
Container Component:	-
Arguments:	Model name
Return:	
Pre-Condition:	Specific model to be tuned
Post-Condition:	Best set of parameter for the particular model.
Exception:	-

Table 3.9 The models are in default trained using the default parameters. But in order to increase the model's performance, the parameters can be adjusted. The portal provides facilities to gives parameter and trains the models on that parameters and hyper-parameter tuning on a set of parameters can also be done.

Table 3.10 Visualization

Name:	Hyper Parameter Tuning
File Name:	Tune
Naming Convention:	-
Short Description:	This page is for tune the default parameter. To increase the performance of the algorithm.
Container Component:	-
Arguments:	Model name
Return:	
Pre-Condition:	Specific model to be tuned
Post-Condition:	Best set of parameter for the particular model.
Exception:	-

Table 3.10 The visualization part is also two types. i.e. the data visualization before model building on which the mean, median, mode, max, min and outliers of each attribute are displayed.

And then after model training, the confusion matrix, ROC curve and ground truth and predicted values and plotted in a graph and displayed.

CHAPTER 4

SYSTEM IMPLEMENTATION

4.1 Abstract Code

Module – 1 HomePage Activity

```
Function Uploadfile()
//Input : File from local system
//Output: uploaded file in server
{
        If(format(file) != csv)
                throw Exception
        Else
                 file uploaded in the server
}
```

Module – 2 Pre-Processing

```
Function Preprocess()
//Input : Uploaded file
//Output : Preprocessed Data
{
        Display the available attributes in different rows
        For each row in the table:
                If missing value = NaN:
                        If replace_with = mean
                                Replace_by = mean(attribute)
                        If replace_with = median
                                Replace_by = median(attribute)
                        If replace_with = mode
                                Replace_by = mode(attribute)
                        If replace_with = custom
                                Replace_by = custom_value
```

```
                    Replace all NaN values in the particular attribute with
replace_by value
                If missing value = ?:
                    If replace_with = mean
                        Replace_by = mean(attribute)
                    If replace_with = median
                        Replace_by = median(attribute)
                    If replace_with = mode
                        Replace_by = mode(attribute)
                    If replace_with = custom
                        Replace_by = custom_value
                    Replace all ? values in the particular attribute with
replace_by value
                If missing value = custom:
                    If replace_with = mean
                        Replace_by = mean(attribute)
                    If replace_with = median
                        Replace_by = median(attribute)
                    If replace_with = mode
                        Replace_by = mode(attribute)
                    If replace_with = custom
                        Replace_by = custom_value
                    Replace all rows with the given custom value in the
particular attribute with replace_by value

    }
```

Module – 3 Model Training

```
Function trainModel()
//Input : Preprocessed Dataset
//Output: Trained Model, F1-score
{
    Model = model selected by the user
    Split_ratio = test-train split given by the user.
    Train the model and display the train and test F1 scores
}
```

Module – 4 Hyper parameter tuning

```
Function hyperParameterTuning()
//Input : Model Architecture with set of parameters
//Output : Best set of parameters
{
        Based on the parameters specified by the user the model is tuned

        If method == GridSearchCV
                Perform Grid Sraech CV operation will the given set of parameters in the model selected and return the best set of parameters.

        If method == k-fold cross validation
                Cross validation is done and the model whose performance is much better than the one before is returned.
}
```

Module – 5 Visualization

```
Function visualize()
//Input : dataset
//Output : Visualized image of dataset
{
        Data are plotted with the help of matplotlib and,seaborn libraries
        The confusion matrix is plotted in terms of classification
        The plot between actual and predicted values is plotted in case of regression
        For data visualization box plots and scatter plots are drawn
}
```

Module – 6 Model export

```
Function test()
//Input : trained model
//Output : model's pickle file
```

```
{
        Model object is converted to string and dumped in to the named model's
        Pickle file
}
```

4.2 SCREENSHOTS

Home page

Figure 4.1 shows the home page of the web application for fastml . This page display the application name and ask for the input file and the file should be in csv format.

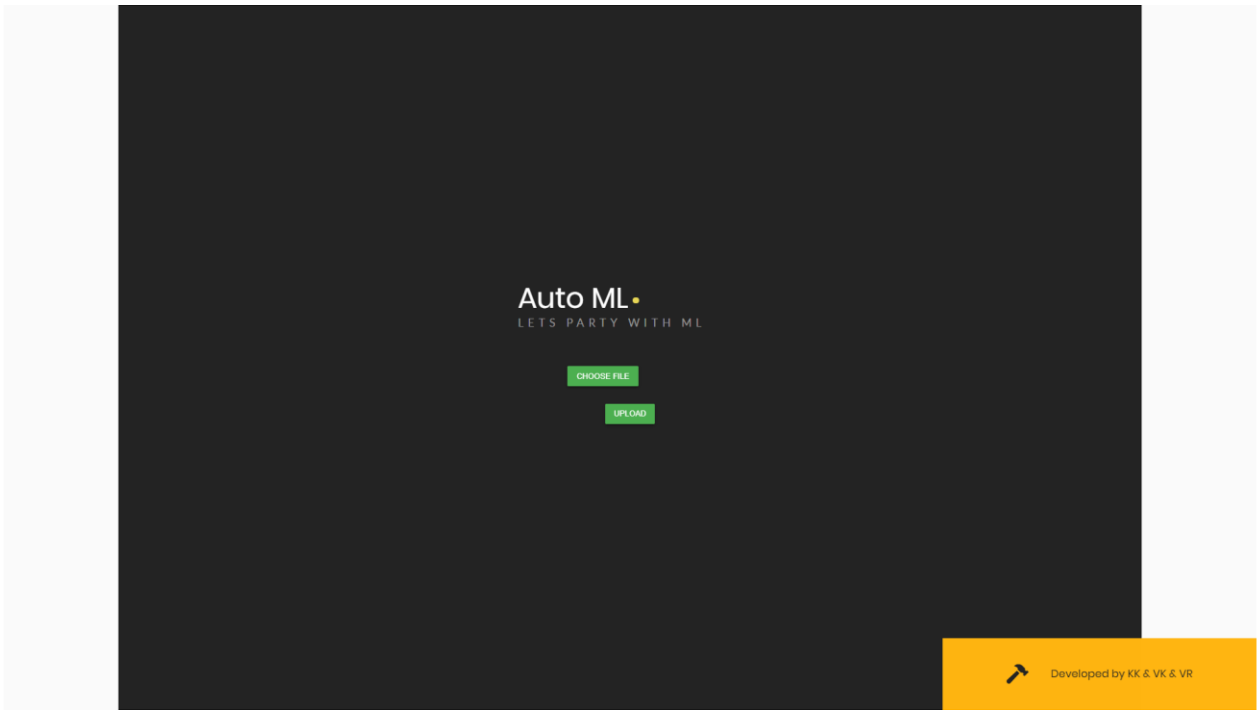
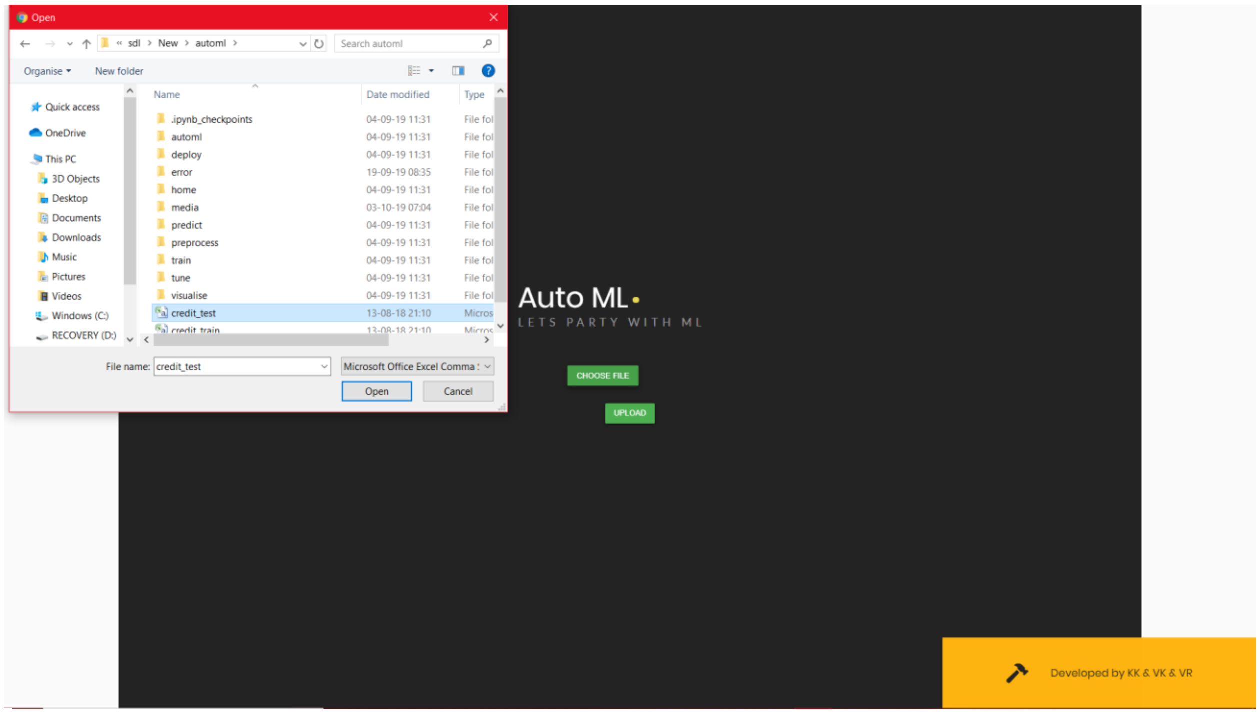


Figure 4.1 Home page



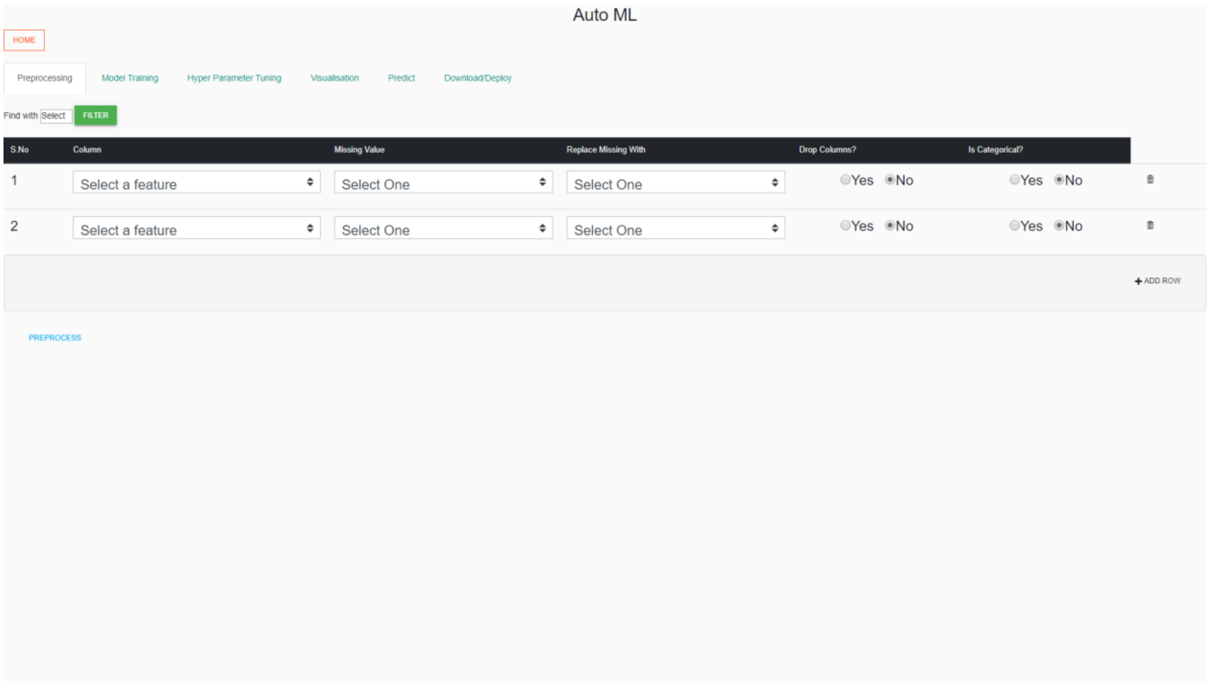
Name	Date modified	Type
.ipynb_checkpoints	04-09-19 11:31	File fol
automl	04-09-19 11:31	File fol
deploy	04-09-19 11:31	File fol
error	19-09-19 08:35	File fol
home	04-09-19 11:31	File fol
media	03-10-19 07:04	File fol
predict	04-09-19 11:31	File fol
preprocess	04-09-19 11:31	File fol
train	04-09-19 11:31	File fol
tune	04-09-19 11:31	File fol
visualise	04-09-19 11:31	File fol
credit_test	13-08-18 21:10	Micros
credit_train	13-08-18 21:10	Micros

File name: credit_test
Microsoft Office Excel Comma
Open Cancel

Figure 4.2 File upload

Preprocessing

Figure 4.3 shows the pre-processing page. The page shows the all the features in the dataset. Users have to choice whether the feature have to be drop or not and it shows whether the class label is categorical or not. And replace the missing value also with mean , median or with custom input.

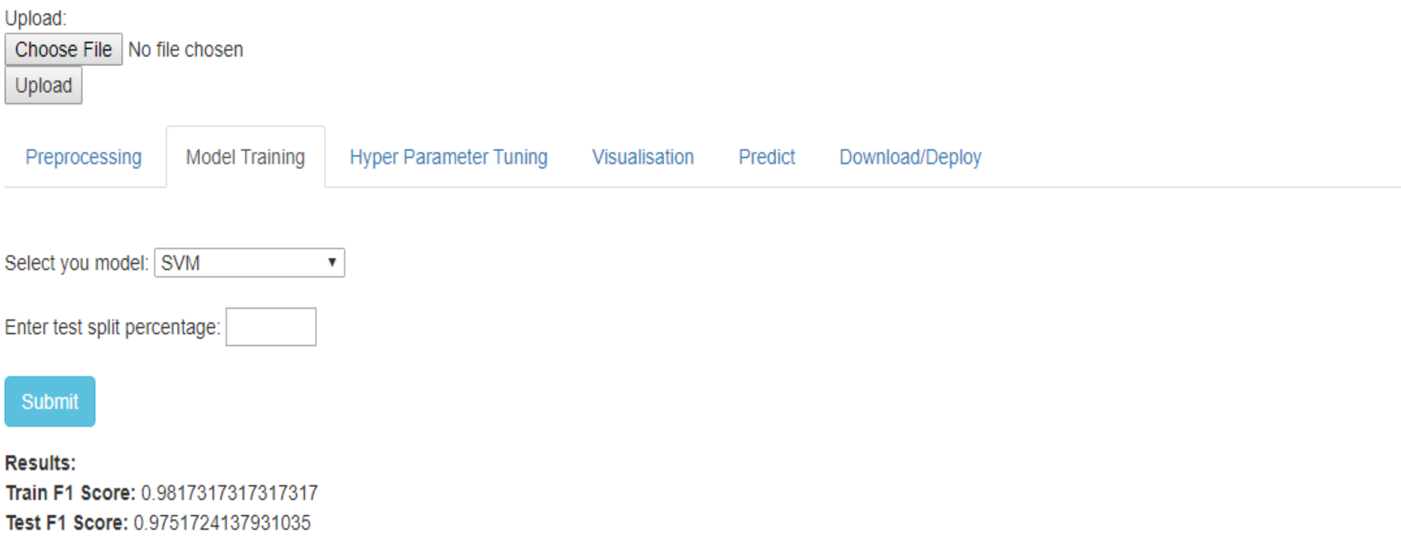


S.No	Column	Missing Value	Replace Missing With	Drop Columns?	Is Categorical?
1	Select a feature	Select One	Select One	Yes [✓ No]	Yes [✓ No]
2	Select a feature	Select One	Select One	Yes [✓ No]	Yes [✓ No]

Figure 4.3 pre-processing

Model Training

After preprocessing stage, training stage, where based on the dataset uploaded it is identified whether it’s an regression or classification task and only those models are displayed. The regression models available are Linear Regression, Lasso Regression and Ridge Regression etc. The classification models included in the site are SVM, Decision Tree, Logistic Regression, Naïve Bayesian and knn. Then the split ratio has to be selected. After clicking on the train button, the model will be trained on the default set of parameters and the given split ratio and the resulting metric scores are returned.



Select you model: SVM

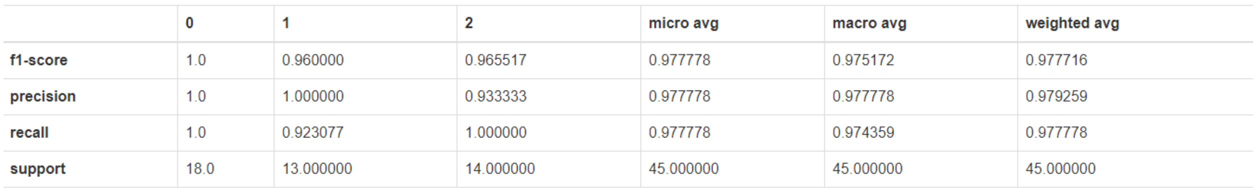
Enter test split percentage:

Results:
Train F1 Score: 0.9817317317317317
Test F1 Score: 0.9751724137931035

Figure 4.4 shows the Model Training

Confusion Matrix

A **confusion matrix** is a table that is often used to describe the performance of a classification model (or “classifier”) on a set of test data for which the true values are known. It allows the visualization of the performance of an algorithm**.**



	0	1	2	micro avg	macro avg	weighted avg
f1-score	1.0	0.960000	0.965517	0.977778	0.975172	0.977716
precision	1.0	1.000000	0.933333	0.977778	0.977778	0.979259
recall	1.0	0.923077	1.000000	0.977778	0.974359	0.977778
support	18.0	13.000000	14.000000	45.000000	45.000000	45.000000

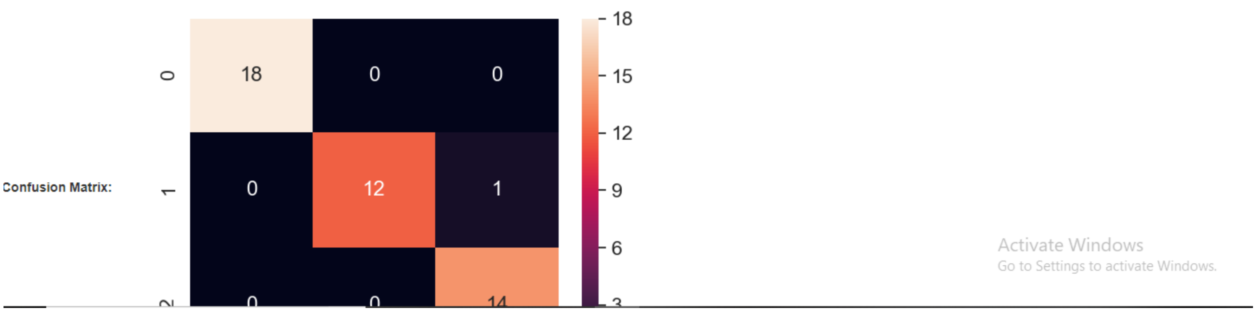
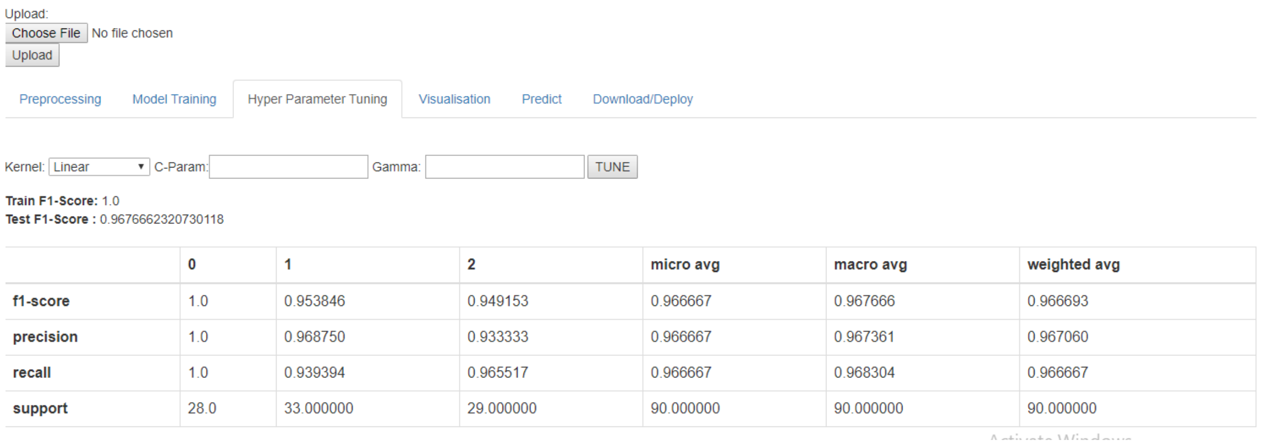


Figure 4.5 shows the Confusion Matrix

Hyper parameter tuning

The models are in default trained using the default parameters. But in order to increase the model's performance, the parameters can be adjusted. The portal provides facilities to gives parameter and trains the models on that parameters and hyper-parameter tuning on a set of parameters can also be done.



Kernel: Linear C-Param: Gamma:

Train F1-Score: 1.0
Test F1-Score : 0.9676662320730118

	0	1	2	micro avg	macro avg	weighted avg
f1-score	1.0	0.953846	0.949153	0.966667	0.967666	0.966693
precision	1.0	0.968750	0.933333	0.966667	0.967361	0.967060
recall	1.0	0.939394	0.965517	0.966667	0.968304	0.966667
support	28.0	33.000000	29.000000	90.000000	90.000000	90.000000

Figure 4.6 shows the Hyper parameter Tuning

Visualization

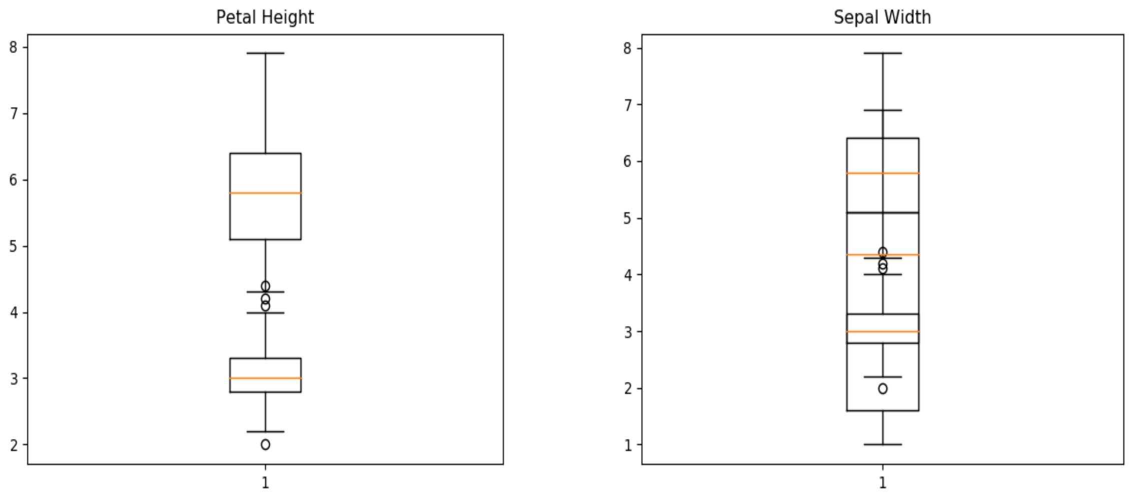


Figure 4.7 shows the Box plot

The visualization part is also two types. i.e. the data visualization before model building on which the mean, median, mode, max, min and outliers of each attribute are displayed.

And then after model training, the confusion matrix, ROC curve and ground truth and predicted values and plotted in a graph and displayed.

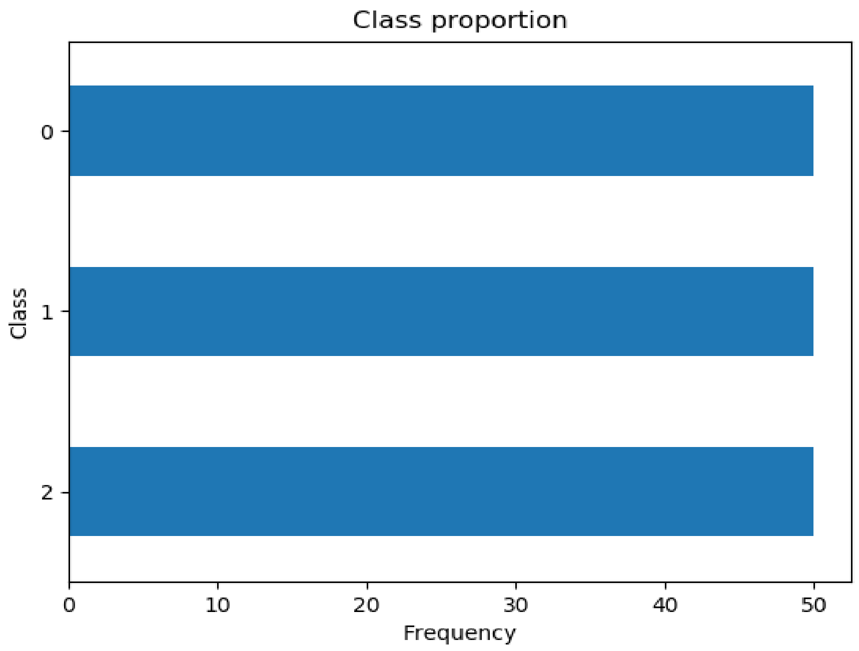
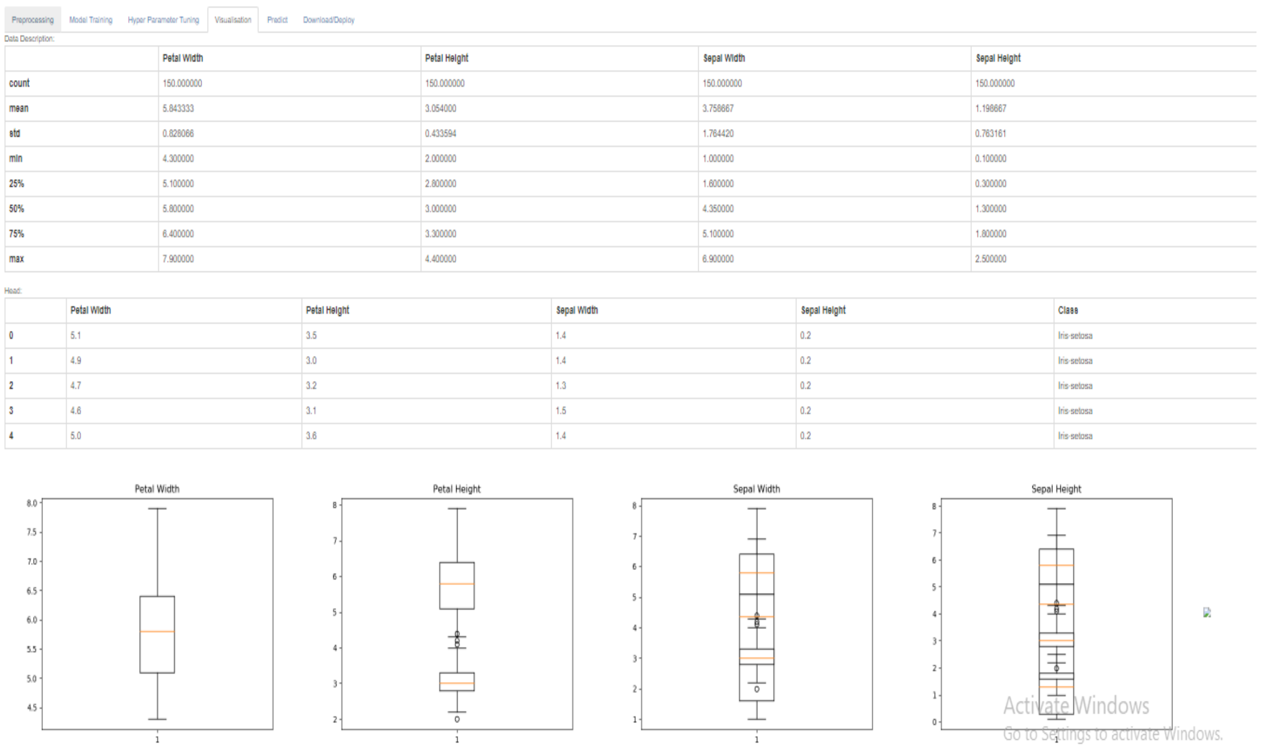


Figure 4.8 shows the Class Proportion



Data Description:

	Petal Width	Petal Height	Sepal Width	Sepal Height
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

Head:

	Petal Width	Petal Height	Sepal Width	Sepal Height	Class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

Figure 4.9 Shows The visualization page

CHAPTER 5

TEST CASES AND TEST RESULTS

5.1 TEST CASES AND RESULT

Table 5.1 Test cases and results for File upload

Test case Id	T1
Test case description	The user should upload the file for further process
Test data	Click upload button and chose the file from your local system
Expected output	Pre-processing page is displayed
Test result	PASS

Table 5.2 Test cases and results for File Upload

Test case Id	T2
Test case description	The user not select the file but click the upload button
Test data	Click upload button and not chose the file from your local system
Expected output	Error Handling page is displayed
Test result	PASS

Table 5.3 Test cases and results for Pre-processing

Test case Id	T3
Test case description	Users have to choice whether the feature have to be drop or not and it shows whether the class label is categorical or not. And replace the missing value also with mean , median or with custom input.
Test data	Click pre-process button after fill the needed requirements.
Expected output	Pre-processed Dataset
Test result	PASS

Table 5.4 Test cases and results for Model development

Test case Id	T4
Test case description	Model are trained with default parameters
Test data	Select Split percentage.
Expected output	Model Trained
Test result	PASS

Table 5.5 Test cases and results for Model development

Test case Id	T5
Test case description	Model are trained with default parameters
Test data	Select Split percentage equals to Zero
Expected output	Throw Exception
Test result	PASS

Table 5.6 Test cases and results for Hyper parameter tuning

Test case Id	T6
Test case description	Best Parameter are Selected
Test data	Parameter should be 0 to 1.
Expected output	Best Parameter are selected and returned
Test result	PASS

Table 5.7 Test cases and results for Visualization

Test case Id	T7
Test case description	Data are plotted with the help of matplotlib and seaborn
Test data	Processed Data set
Expected output	Plotted images are produced
Test result	PASS

Table 5.8 Test cases and results for Visualization

Test case Id	T8
Test case description	Data are plotted with the help of matplotlib and seaborn
Test data	Not pre-processed data set
Expected output	Plotted images are produced
Test result	PASS

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT

Conclusion

Easy development of Machine learning model through our Website Tool .Easy to deploy a model and user need not worry about programming as they are Automated. Thus our project serves as a platform for automatic Machine Learning training. It makes the job of data scientists and researchers more easy. It could increase the pace of training and tuning models using our platform. The users can download model object pickle file after training.

This system provide UI tool for developing machine learning without involving in programming. The Web application is mainly designed for the purpose to be used by Mathematicians, Data Scientists and other non-programmers and they can train and export their Machine Learning models without a single line of coding. The entire process of preprocessing, model training, visualization, hyper parameter tuning and model export can be done using the web application without doing a single line of code.

Future Enhancement

The user who doesn’t know much about the models and mathematics, statistics and probability can just upload their datasets and the site will do everything automated and finally gives insights on the data and deploys it as a web service and provides the service key and url to be called as a REST API. So our future enhancement will be predict model is suitable for which dataset and automatic pre-processing also. The datasets can be uploaded as csv files directly or Kaggle API can be provided or the bucket from GCP or AWS can be authenticated and datasets from them can also be loaded.

APPENDIX I

Code

Home -> views.py

```
from django.shortcuts import render
from sklearn import svm
import numpy as np
import pandas as pd
from sklearn.preprocessing import LabelEncoder

def index(request):
    return render(request, 'home/index.html')

def preprocess(request):
    csv_file = request.FILES["myfile"]
    file_data = csv_file.read().decode("utf-8")
    lines = file_data.split("\n")
    df = []
    field_names = lines[0].split(",")

    for line in lines[1:]:
        fields = line.split(",")
        if(fields[0] != ''):
            df.append(fields)

    head = df[0]
    encoder = []
    for i in range(0,len(head)):
        if(head[i][0].isalpha()):
            encoder.append(i)

    df = pd.DataFrame(df)
    label_encoder = LabelEncoder()
    for i in encoder:
        df.iloc[:,i]= label_encoder.fit_transform(df.iloc[:,i])

    clf = svm.SVC(gamma='scale')
    clf.fit(df.iloc[:,:-1], df.iloc[:,-1])
    return render(request, 'home/index.html')
```

Preprocess -> views.py

```
from django.shortcuts import render, redirect
from django.core.files.storage import FileSystemStorage
import os
from automl import settings
import pandas as pd
import numpy as np
import statistics as st
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import Imputer

def index(request):
    myfile = request.FILES['myfile']
    file_data = myfile.read().decode("utf-8")
    lines = file_data.split("\n")
    field_names = lines[0].split(",")
    if(myfile):
        request.session['data_set'] = myfile.name
        fs = FileSystemStorage()
        path = os.path.join(settings.BASE_DIR,'media')
        fs = FileSystemStorage(location=f'{path}')
        filename = fs.save(myfile.name, myfile)
        PROJECT_ROOT = os.path.join(settings.BASE_DIR,'media')
        filename = os.path.join(PROJECT_ROOT,f'{filename}')
    context = {
        'headers':field_names,
        'file_name':myfile.name,
    }
    return render(request, 'preprocess/index.html', context)



def preprocess(request):
    if(request.method == 'POST'):
        feature = request.POST.getlist('feature')
        featuress = []
        for i in feature[:-1]:
            featuress.append(i)
        if(feature[-1][-1] == '\n'):
            featuress.append(feature[-1][:-2])
        else:
            featuress.append(feature[-1])
        missing = request.POST.getlist('missing')
        replace = request.POST.getlist('replace')
        is_drop = []
        is_categorical = []
        for key in request.POST:
            if('drop' in key):
                is_drop.append(request.POST.getlist(key)[0])
            if('categorical' in key):
                is_categorical.append(request.POST.getlist(key)[0])
        missing_custom = request.POST.getlist('missing_custom')
        replace_custom = request.POST.getlist('replace_custom')
        df = pd.read_csv(r'media/'+request.session['data_set'])
        features = {}
        m = 0
        n = 0
        for i in range(len(featuress)):
            temp = [missing[i], replace[i], is_drop[i], is_categorical[i]]
            if(temp[0] == '3'):
                temp.append(missing_custom[m])
                m+=1
            if(temp[1] == '3'):
                temp.append(replace_custom[n])
                n+=1
            features[featuress[i]] =  temp
        p_type = "classification"
        if(df.iloc[:,-1].nunique()/df.iloc[:,-1].count() > 0.05):
            p_type="regression"
        request.session['type'] = p_type

        for i in features:
            todo = features[i]
            #drop
            if(todo[2]=='1'):
                del df[i]
            else:
                #continuous
                if todo[3] == '2':
                    #missing = 'Nan'
                    if todo[0] == '2':
                        #replace with 'mean'
                        if todo[1] == '1':
                            imputer = Imputer(missing_values = 'NaN', strategy = 'mean', axis = 0)
                            imputer = imputer.fit(np.array(df[i]).reshape(-1,1))
                            s = imputer.transform(np.array(df[i]).reshape(-1,1))
                            df[i] = s.ravel()
                        #replace with median
                        elif todo[1] == '2':
                            imputer = Imputer(missing_values = 'NaN', strategy = 'median', axis = 
        0)

                            imputer = imputer.fit(np.array(df[i]).reshape(-1,1))
                            s = imputer.transform(np.array(df[i]).reshape(-1,1))
                            df[i] = s.ravel()
                        #replace with custom replace
                        elif todo[1] == '3':
                            for n,j in enumerate(df[i]):
                                if(np.isnan(j)):
                                    df[i][n] = todo[-1]
            else:
                print("Nothing")
        #missing = '?'
        elif todo[0] == '1':
            #replace with 'mean'
            if todo[1] =='1':
                repl = st.mean(df[i])
                for n,j in enumerate(df[i]):
                        if(j=="?"):
                            df[i][n] = repl
            #replace with 'mediam'
            elif todo[1] =='2':
                repl = st.median(df[i])
                for n,j in enumerate(df[i]):
                    if(j=="?"):
                        df[i][n] = repl
            #replace with custom replace
            elif todo[1] =='3':
                for n,j in enumerate(df[i]):
                    if(j=='?'):
                        df[i][n] = todo[-1]
            else:
                print("no replace")
        #custom missing
        elif todo[0] == '3':
            #replace with 'mean'
            if todo[1] =='1':
                repl = st.mean(df[i])
                for n,j in enumerate(df[i]):
                        if(j==todo[-2]):
                            df[i][n] = repl
            #replace with 'mediam'
            elif todo[1] =='2':
                repl = st.median(df[i])
                for n,j in enumerate(df[i]):
                        if(j==todo[-2]):
                            df[i][n] = repl
                #replace with custom replace
                elif todo[1] =='3':
                    for n,j in enumerate(df[i]):
                        if(j==todo[-2]):
                            df[i][n] = todo[-1]
                else:
                    print("no replace")
        else:
            print('categorical')
            #missing = 'Nan'
            if todo[0] == '2':
                #replace with 'most frequent'
                if todo[1] == '1':
                    imputer = Imputer(missing_values = 'NaN', strategy = 'most_frequent',
        axis = 0)
                    imputer = imputer.fit(np.array(df[i]).reshape(-1,1))
                    s = imputer.transform(np.array(df[i]).reshape(-1,1))
                    df[i] = s.ravel()
                #replace with custom replace
                elif todo[1] == '2':
                    for n,j in enumerate(df[i]):
                        if(np.isnan(j)):
                            df[i][n] = todo[-1]
                else:
                    print("Nothing")
            #missing = '?'
            elif todo[0] == '1':
                #replace with 'mode'
                if todo[1] =='1':
                    repl = st.mode(df[i])
                    for n,j in enumerate(df[i]):
                            if(j=="?"):
                                df[i][n] = repl
                                #replace with custom replace
                                elif todo[1] =='2':
                                    for n,j in enumerate(df[i]):
                                        if(j=='?'):
                                            df[i][n] = todo[-1]
                                else:
                                    print("no replace")
                            #custom missing
                            elif todo[0] == '3':
                                #replace with 'mode'
                                if todo[1] =='1':
                                    repl = st.mode(df[i])
                                    for n,j in enumerate(df[i]):
                                            if(j==todo[-2]):
                                                df[i][n] = repl
                                #replace with custom replace
                                elif todo[1] =='2':
                                    for n,j in enumerate(df[i]):
                                        if(j==todo[-2]):
                                            df[i][n] = todo[-1]
                                else:
                                    print("no replace")
                            else:
                                print("nothing")
                            #encode
                            le = LabelEncoder()
                            le = le.fit(np.array(df[i]).reshape(-1,1))
                            s = le.transform(np.array(df[i]).reshape(-1,1))
                            df[i] = s.ravel()
            df.to_csv(r'media/'+request.session['data_set'], index=False, header=True)
            return redirect('/train')
```

Train -> Views.py

```
from django.shortcuts import render
from sklearn.model_selection import train_test_split
from sklearn import svm
import pandas as pd
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn import linear_model
from sklearn.metrics import r2_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
import seaborn as sn

def index(request):
    classification_models = ["SVM", "KNN", "Decision Tree", "Logistic Regression"]
    regression_models = ["Linear Regression", "Lasso Regression", "Ridge Regression",
"Bayesian Ridge Regression"]
    models = regression_models
    if(request.session["type"]=="classification"):
        models = classification_models
    context = {
        'models': models,
    }
    return render(request, 'train/index.html', context)



def metrics(y_true,y_pred):
    from sklearn.metrics import f1_score
    f1 = f1_score(y_true, y_pred, average='macro')
    return f1

def train(request):
    df = pd.read_csv(r'media/'+request.session['data_set'])
    X = df.iloc[:,:-1]
    y = df.iloc[:,-1]
```

```
        X_train,X_test,y_train,y_test = train_test_split(X, y,
test_size=int(request.POST['split'])/100)
        model = request.POST['model']

        classification_models = ["SVM", "KNN", "Decision Tree", "Logistic Regression"]
        regression_models = ["Linear Regression", "Lasso Regression", "Ridge Regression",
"Bayesian Ridge Regression"]
        models = regression_models
        if(request.session['type']=="classification"):
          models = classification_models
        request.session['model'] = model

        if model =='SVM':
          clf = svm.SVC(C=1.0, kernel='linear')

        elif model== 'KNN':
          clf = KNeighborsClassifier(n_neighbors=5)

        elif model == 'Decision Tree':
          clf = DecisionTreeClassifier(random_state=0)

        elif model == 'Logistic Regression':
          clf = LogisticRegression(random_state=0,
solver='lbfgs',multi_class='multinomial').fit(X_train, y_train)

        elif model == 'Linear Regression':
          reg = linear_model.LinearRegression().fit(X_train, y_train)

        elif model == 'Ridge Regression':
          reg = linear_model.Ridge(alpha=.5).fit(X_train, y_train)

        elif model == 'Lasso Regression':
          reg = linear_model.Lasso(alpha=0.1).fit(X_train, y_train)

        elif models == 'Bayesian Ridge Regression':
          reg = linear_model.BayesianRidge().fit(X_train, y_train)

        if(request.session['type'] == 'classification'):
          clf.fit(X_train, y_train)
          train_pred = clf.predict(X_train)
          train_f1 = str(metrics(y_train,train_pred))
          #plot_prediction(X_train,y_train,train_pred,'Training')
          test_pred = clf.predict(X_test)
          test_f1 = str(metrics(y_test,test_pred))
          #plot_prediction(X_test,y_test,test_pred,'Test')
          report = pd.DataFrame(classification_report(y_test, test_pred,
output_dict=True)).to_html(classes="table table-bordered")
          cm = confusion_matrix(y_test, test_pred)
          df_cm = pd.DataFrame(cm)
          sn.set(font_scale=1.4)
          sns_plot = sn.heatmap(df_cm, annot=True,annot_kws={"size": 16})
          fig = sns_plot.get_figure()
          fig.savefig(r'train/static/images/output.png')

        elif(request.session['type'] == 'regression'):
          train_pred = reg.predict(X_train)
          train_f1 = str(r2_score(y_train,train_pred))
          #plot_prediction(X_train,y_train,train_pred,'Training')
          test_pred = reg.predict(X_test)
          test_f1 = str(r2_score(y_test,test_pred))
          #plot_prediction(X_test,y_test,test_pred,'Test')

        context = {
                'train_f1': train_f1,
                'test_f1': test_f1,
                'models': models,
                'clf_report': report,
                'type': request.session['type'],
          }
        return render(request, 'train/index.html', context)
```

Tune -> Views.py

```
from django.shortcuts import render
from sklearn import svm
import pandas as pd
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn import linear_model
from sklearn.metrics import r2_score
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split

def index(request):
    context = {
        'model': request.session['model']
    }
    return render(request, 'tune/index.html', context)

def metrics(y_true,y_pred):
    from sklearn.metrics import f1_score
    f1 = f1_score(y_true, y_pred, average='macro')
    return f1

def tune(request):
    model = request.session['model']
    if(model == 'SVM'):
        clf = svm.SVC(kernel = request.POST['kernel'], C = float(request.POST['cparam']),
gamma=float(request.POST['gamma']))
    elif model == 'KNN':
        clf = KNeighborsClassifier(n_neighbors=request.POST['k'])
    elif model == 'Decision Tree':
        clf = DecisionTreeClassifier(random_state=0, criterion = request.POST['criterion'])
    elif model == 'Logistic Regression':
        clf = LogisticRegression(C = request.POST['cparam'], max_iter =
request.POST['iterations'])
    elif model == 'Linear Regression':
        reg = linear_model.LinearRegression()

    df = pd.read_csv(r'media/'+request.session['data_set'])
    X = df.iloc[:,:-1]
    y = df.iloc[:,-1]
    X_train,X_test,y_train,y_test = train_test_split(X, y, test_size=0.6)

    if(request.session['type'] == 'classification'):
        clf.fit(X_train, y_train)
        train_pred = clf.predict(X_train)
        train_f1 = str(metrics(y_train,train_pred))
        #plot_prediction(X_train,y_train,train_pred,'Training')
        test_pred = clf.predict(X_test)
        test_f1 = str(metrics(y_test,test_pred))
        #plot_prediction(X_test,y_test,test_pred,'Test')
        report = pd.DataFrame(classification_report(y_test, test_pred,
output_dict=True)).to_html(classes="table table-bordered")
    else:
        reg.fit(X_train, y_train)
        train_pred = reg.predict(X_train)
        train_f1 = str(r2_score(y_train,train_pred))
        #plot_prediction(X_train,y_train,train_pred,'Training')
        test_pred = reg.predict(X_test)
        test_f1 = str(r2_score(y_test,test_pred))
        #plot_prediction(X_test,y_test,test_pred,'Test')

    context = {
        'model': model,
        'train_f1': train_f1,
        'test_f1': test_f1,
        'clf_report': report,
        'type': request.session['type'],
    }
    return render(request, 'tune/index.html', context)
```

Visualize -> views.py

```
from django.shortcuts import render
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn import svm
import matplotlib.pyplot as plt
import matplotlib.pyplot as plt1
import seaborn as sns

def index(request):
    df = pd.read_csv(r'media/'+request.session['data_set'])
    desc = df.describe().to_html(classes="table table-bordered")
    head = df.head().to_html(classes="table table-bordered")
    d = df.iloc[:,:-1]
    for i in range(len(d.columns)):
        plt1.title(d.columns[i])
        plt1.boxplot(d[d.columns[i]])
        plt1.savefig(r'visualise/static/images/'+'box'+str(i)+'.png')
    box_plot_images = []
    plt1.close()
    plt.title("Class proportion")
    plt.ylabel("Class")
    plt.xlabel("Frequency")
    df.iloc[:,-1].value_counts().plot( kind='barh')
    plt.savefig(r'visualise/static/images/graph.png')
    for i in range(len(df.columns)):
        box_plot_images.append("images/box"+str(i)+".png")

    context = {
        'desc': desc,
        'head': head,
        'box_plot_images': box_plot_images,
    }
    return render(request, 'visualise/index.html', context)
```

RapidML -> urls.py

```
from django.contrib import admin
from django.urls import path
from django.conf.urls import url, include
from automl import settings
from django.conf.urls.static import static

urlpatterns = [
    url('home/', include('home.urls')),
    url('train/', include('train.urls')),
    url('deploy/', include('deploy.urls')),
    url('predict/', include('predict.urls')),
    url('preprocess/', include('preprocess.urls')),
    url('tune/', include('tune.urls')),
    url('visualise/', include('visualise.urls')),
    url('^$', include('home.urls')),
    path('admin/', admin.site.urls),
]+static(settings.STATIC_URL, document_root=settings.STATIC_ROOT)
```

Preprocess -> index.html

```
<html>
    <head>
      <title>RapidML</title>
      <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css">
      <script src="http://code.jquery.com/jquery-latest.js"></script>
      <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
      <script>
        jQuery(document).delegate('a.add-record', 'click', function(e) {
            e.preventDefault();
            var content = jQuery('#sample_table tr'),
    size = jQuery('#tbl_posts >tbody >tr').length + 1,
    element = null,
    element = content.clone();
    element.attr('id', 'rec-'+size);
    element.find('.drop').attr('name', 'drop'+size);
    element.find('.categorical').attr('name', 'categorical'+size);
    element.find('.delete-record').attr('data-id', size);
    element.appendTo('#tbl_posts_body');
    element.find('.sn').html(size);
  });

jQuery(document).delegate('a.delete-record', 'click', function(e) {
    e.preventDefault();
    var didConfirm = confirm("Are you sure You want to delete");
    if (didConfirm == true) {
      var id = jQuery(this).attr('data-id');
      var targetDiv = jQuery(this).attr('targetDiv');
      jQuery('#rec-' + id).remove();

    $('#tbl_posts_body tr').each(function(index) {
      $(this).find('span.sn').html(index+1);
    });
    return true;
  } else {
    return false;
  }
});

jQuery(document).delegate('td.select-option', 'change', function(e) {
    e.preventDefault();
    var selectValue = $(this).find('select');
    var textBox = $(this).find('input')
    if(selectValue['0'].value == '3'){
        textBox['0'].style.display = "block"
    }
```

```
        else {
            textBox['0'].style.display = "none";
        }
    });
    </script>
</head>
<body>
  <center>
    <h2> Auto ML </h2>
  </center>
  <form method="post" enctype="multipart/form-data" action="/preprocess/">
    {% csrf_token %}
    <br>
    Upload: <input type="file" name="myfile">
    <button type="submit">Upload</button>
  </form>
  <ul class="nav nav-tabs">
    <li class="active"><a href="/preprocess">Preprocessing</a></li>
    <li><a href="/train">Model Training</a></li>
    <li><a href="/tune">Hyper Parameter Tuning</a></li>
    <li><a href="/visualise">Visualisation</a></li>
    <li><a href="/predict">Predict</a></li>
    <li><a href="/deploy">Download/Deploy</a></li>
            </ul>
<form method="post" >
    {% csrf_token %}
    <table class="table table-bordered" id="tbl_posts">
    <thead>
      <tr>
        <th>S.No</th>
        <th>Column</th>
        <th>Missing Value</th>
        <th>Replace Missing With</th>
        <th>Drop Columns?</th>
        <th>Is Categorical?</th>
```

```
           </tr>
          </thead>
          <tbody id="tbl_posts_body">
           <tr id="rec-1">
            <td><span class="sn">1</span></td>
            <td class="ui-widget">
             <select name="feature">
              <option value="0" selected>Select a feature</option>
               {% for header in headers %}
                <option value="{{header}}">{{header}}</option>
               {% endfor %}
             </select>
            </td>
            <td class="ui-widget select-option"><select class="missing" name="missing">
              <option value="0" selected>Select One</option>
              <option value="1">?</option>
              <option value="2">NaN</option>
              <option value="3">Custom Input</option>
             </select>
             <input type="text" name="missing_custom" id="missing_custom"
style='display:none;'/>
            </td>
            <td class="ui-widget select-option"><select class="replace" name="replace">
              <option value="0" selected>Select One</option>
              <option value="1">Mean</option>
              <option value="2">Median</option>
              <option value="3">Custom Input</option>
             </select>
             <input type="text" name="replace_custom" class='color' id="color"
style='display:none;'/>
            </td>
            <td class="ui-widget"><center><input type="radio" name="drop"
value="1"/>Yes&nbsp;&nbsp;&nbsp;<input type="radio" name="drop" value="2"
checked/>No</center></td>
```

```
                        <td class="ui-widget"><center><input class="categorical" type="radio"
name="categorical" value="1"/>Yes&nbsp;&nbsp;&nbsp;<input type="radio"
class="categorical" name="categorical" value="2" checked/>No</center></td>
                        <td><a class="btn btn-primary delete-record" data-id="1"><i class="glyphicon
glyphicon-trash"></i></a></td>
                      </tr>
                    </tbody>
                  </table>
                  <div class="well clearfix">
                        <a class="btn btn-primary pull-right add-record" data-added="0"><i
class="glyphicon glyphicon-plus"></i> Add Row</a>
                  </div>
                    &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<input type="submit"
value="Preprocess" onclick="javascript: form.action='dopreprocess';" class="btn btn-info"/>
                </form>

              <div style="display:none;">
                <table id="sample_table">
                  <tr id="">
                    <td><span class="sn"></span></td>
                        <td class="ui-widget">
                          <select name="feature" class"feature">
                            <option value="0" selected>Select a feature</option>
                              {% for header in headers %}
                                <option value="{{header}}">{{header}}</option>
                              {% endfor %}
                          </select>
                        </td>
                        <td class="ui-widget select-option"><select class="missing" name="missing">
                            <option value="0" selected>Select One</option>
                            <option value="1">?</option>
                            <option value="2">NaN</option>
                            <option value="3">Custom Input</option>
                          </select>
```

```
                       <input type="text" name="missing_custom" id="missing_custom"
style='display:none;'/>
                     </td>
                     <td class="ui-widget select-option"><select class="replace" name="replace">
                         <option value="0" selected>Select One</option>
                         <option value="1">Mean</option>
                         <option value="2">Median</option>
                         <option value="3">Custom Input</option>
                       </select>
                       <input type="text" name="replace_custom" id="color" style='display:none;'/>
                     </td>
                     <td class="ui-widget"><center><input class="drop" type="radio" name=""
value="1"/>Yes&nbsp;&nbsp;&nbsp;<input type="radio" class="drop" name="" value="2"
checked/>No</center></td>
                     <td class="ui-widget"><center><input class="categorical" type="radio" name=""
value="1"/>Yes&nbsp;&nbsp;&nbsp;<input type="radio" class="categorical" name=""
value="2" checked/>No</center></td>
                     <td><a class="btn btn-primary delete-record" data-id="0"><i class="glyphicon
glyphicon-trash"></i></a></td></tr></table> </div></body></html>
```