UnFold It

Automated Machine Learning Web-App
https://github.com/biplavadhikary/AutoML

Project Report
2.0

May 30th, 2020

Submitted in full requirements of
Minor Project [IT – 3082]
Instructor: Ms. Divya Kumari [Assistant Professor]

Team: The Unfolders

Roll Number	Name	Position	Contribution
1706119	Avinash Soni	Team Leader	Web Designing and creation of Home-Page and Team Management
1706122	Ayush Agrawal	Team Member	ML Specialization, generation of required models and scripts and creation of useful content
1706124	Biplav Adhikary	Team Member	Web Developer, Server-Side Coding, handling Dynamic Pages, implementing scripts and Project Integration
1706130	Gulam Muhammad	Team Member	Content and Bot Creator and also a key part in Documentation handling

School of Computer Engineering
Kalinga Institute of Industrial Technology
Deemed to be University
Bhubaneswar-751024
Autumn 2019

KIIT Deemed to be University
School of Computer Engineering
Bhubaneswar, ODISHA
751024

CERTIFICATE

This is to certify that the project entitled

UnFold It 'The Automated Machine Learning Web-App'

is a record of Bona fide work carried out by

AVINASH SONI - 1706119
AYUSH AGRAWAL - 1706122
BIPLAV ADHIKARY - 1706124
GULAM MUHAMMAD - 1706130

in the partial fulfilment of the requirement for Minor Project (6th Semester) from Department IT (School of Computer Science and Engineering) at KIIT Deemed to be university, Bhubaneswar. This work is done during year 2019-2020, under the guidance of:

Date: 25 April, 2020

Ms. Divya Kumari
Assistant Professor

ACKNOWLEDGEMENT

We would like to express our profound gratitude to Prof. Divya Kumari Sharma, for her expert guidance and encouragement throughout which helped us in accomplishing this project. We would also like to thank the CS, T&P and CAAS Department for allowing us to conduct the project. We came to know a lot many new things during the course of this project. And therefore, we are really thankful to all of them. This project would not have been accomplished successfully without the cooperation, hard-work and efforts of everyone.

ABSTRACT

With the vast and continuous increase in the amount of data in our world of technology, it has been acknowledged that the number of expert data scientists cannot succumb to address these challenges. Thus, there is a crucial need of automating this process of building expert machine learning models. In the past few years, several frameworks and techniques have been introduced to face the challenge of automating the process of Combined Algorithm Selection and Hyper-parameter-tuning (CASH) in the domain of Machine learning. The main aim of these techniques is to reduce the role of the human in the loop and fill this gap for non-expert machine learning users by playing the role of the domain expert. In this project, we will be presenting a comprehensive survey for the state-of-the-art efforts in tackling the CASH problem along Auto-Visualization. In addition, we will highlight the research work of automating the other steps of full complex machine learning pipeline (AutoML) from data understanding till Model Prediction. Furthermore, we will provide some coverage on the various tools and frameworks that have been introduced in this domain. And finally, we will discuss some of the open challenges and research directions that needs to be addressed in order to achieve this vision and goals of the Automated Machine Learning process.

Keywords

Dataset, Classifier, Visualizer, Automation, Automated ML, Hyper-Parameters, CASH

Table of Contents

1. Introduction	6 - 7
1.1 Purpose	6
1.2 Scope of the Project	6
1.3 Motivation	6
1.4 Related Works	7
2. Objective	8
3. Definitions and Overview	8 - 11
3.1 AutoML Interface	8
3.2 TPOT Classification	9
3.3 AutoViz	10
3.3 Categorical Data Encoding	11
4. Project Details	12 - 18
4.1 Problem Statement	12
4.2 Software Function	12
4.3 Software Perspective	13
4.4 Basic Uses-Cases	14
4.5 Software Design	15
4.6 Project Planning	16
4.2.1 Methodology	16
4.7 Types of Accepted Dataset	16
5. Technologies	19 - 20
5.1 Theme Alignment	19
5.2 Tech Stack	19
5.3 List of Technologies Used	20
6. Implementation	21 - 27
6.1 Data Upload Page	21
6.2 Model Selection Window	21
6.3 Visualization (Plot Generation and Display Window)	22
6.4 Dataset View Window	23
6.5 Accuracy and Customized Download Script Window	23
6.6 Miscellaneous Windows	24
6.7 Project Environment	24
6.8 Automation Script Implementation	26
6.9 Chat-Bot	29
6.9 Home-Page	29
7. Comparison	29
8. Results & Discussion	29 - 30
8.1 Constraints	29
8.2 Assumptions and Dependencies	29
8.3 User Characteristics	30
8.4 Expectations	30
8.5 Tests and Results	31
8.6 Conclusion	32
A. Appendices	33 - 35
B. References	36 - 37

1. Introduction

When we think about technology, the very first thing that comes to our mind is **Machine Learning, AI** etc., which is one of the most trending, demanding & interesting topics of the current tech. world. And as we know, Machine-learning algorithms needs training and statistics to find patterns through volumes of data. Data here, encompasses a load of things— characters, words, numbers, images, clicks, or unorganized data. It powers many of the services that we use today— Recommendation systems that we see on Netflix, YouTube, and Spotify; on search engines like Google and Baidu; social-media feeds like Facebook and Twitter; voice assistants like Alexa and Siri. The list keep going on and on.

So, we thought how would it be if we automate this system of Machine Learning? If we could make algorithms that could be intelligent enough to pick the right algorithm for your given data and train it without much intervention of a person, then we can really save a lot of time and effort. Isn't it interesting? So technically, **Automated machine learning (AutoML)** is the process of automation to apply machine learning to real-world problems.

1.1. Purpose

Data is increasing day by day. Over 2.5 quintillion bytes of data are created every single day, and it's only going to grow from there. But there exist only a selected few people who has the knowledge and skill about to process this data and find out meaningful insight from this data. Suppose a user has sample datasets, but he/she doesn't have any means to process this data, i.e., extract information out of this data and visualize this data properly to understand the data. Sometimes all we want is only to predict the trends from this data, not diving deep into technical stuffs.

1.2. Scope of Project

Any person, be it student, a businessman or someone ranking as high as an IT Engineer who has appropriate data collected with him but lacks the competency to analyze the data, or if he doesn't want to invest time and effort into typing lines of code for simple visualization or prediction can use our Web-App (the project which we have created) to find appropriate insights on his data. This will give him a brief idea and information about his collected data.

1.3. Motivation

In the current few years, the demand for machine learning experts has outpaced the supply, despite the surge of people entering the field. And to address this gap, there have been big stride in the development of user-friendly and simplistic machine learning software that can be used by non-experts of ML. The first steps toward simplifying ML involved developing simple and unified interfaces to a variety of machine learning algorithms (e.g. TPOT, H2O).

Organizations generally face the following issues:

- Availability of expert machine learning developers
- Efficiently scaling trained models to production environment
- Cost

To address this issue, we want to develop a system, where you don't have to invest in a ML expert and waste effort and time in writing lines of code to for simple predictions and visualizations. We, the

UnFolders have come up with a very helpful platform where they just need to have a pre-processed dataset, and just with that, they can perform different actions of their choice through our Web-App.

1.4. Related Works

The following the some works based on the concept of Automated Machine Learning:

- **Eclipse Arbiter (Java):** grid and random search, and genetic search for neural architectures.
- **auto-sklearn:** a drop-in replacement for scikit-learn estimators.
- Auto-WEKA (Java)
- BayesOpt (C++)
- Featuretools: a good library for automatically engineering features from relational and transactional data
- MLBox: distributed data processing, cleaning, formatting, and state-of-the-art algorithms such as LightGBM and XGBoost. It also supports model stacking, which allows you to combine an information ensemble of models to generate a new model aiming to have better performance than the individual models.
- **TPOT:** genetic programming to find the best performing ML pipelines, and it is built on top of scikit-learn
- Hyperopt
- Hyperopt-sklearn
- **Xcessive**: A web-based application for quick, scalable, and automated hyperparameter tuning and stacked ensembling in Python
- H2O
- Advisor
- Spearmint
- BayesianOptimization
- SMAC3
- RoBO
- Scikit-Optimize
- HyperBand
- Optunity
- Cloud AutoML
- ATM
- SigOpt
- DataRobot

2. Objective

We, by means of our project, want to find a way for users to classify, analyze, visualize and predict results from his data by automating all his tasks. All the user has to do is first upload the train dataset, and choose a few options, and later on the test dataset and we'll do everything else for him. Here are the exact steps that the user needs to follow:

▪ Visit our website where you will have the facility to upload any dataset for training first (preferably pre-processed one, we'll go into that later).

▪ After that you have to choose your problem, whether it is a classification or a regression problem or whether you just want to visualize your data.

▪ Next, you'll will have to identify the predictive attributes. With this, the data collection phase will be over.

▪ Now, everything will be done by our Automation Engine. We'll determine the hyper-parameters and the best possible Machine Learning model for your provided dataset. This marks the end of the Modelling phase.

▪ Now all you need to do is just upload the actual dataset whose values need to be predicted or classified.

▪ Using the previously created model, you can now visualize your data and you will be able to see his predictions for the uploaded the test set.

3. Definitions and Overview

3.1 AutoML Interface

The H2O AutoML interface is designed to have as few parameters as possible so that all that a user need to do is point to the dataset, and identify the response column and optionally specify a time constraint or limit on the number of total models trained.

In both the Pyhton and R API, AutoML uses similar data-related arguments, **X, Y, training_frame, validation_frame,** as the other H2O algorithms. Most of the time, all we need to do is specify these data arguments. Then, we can configure values for **max_runtime_secs** and/or **max_models** to set explicitly the time and constrain on number-of-model during the run.

→ Advantages of AutoML:

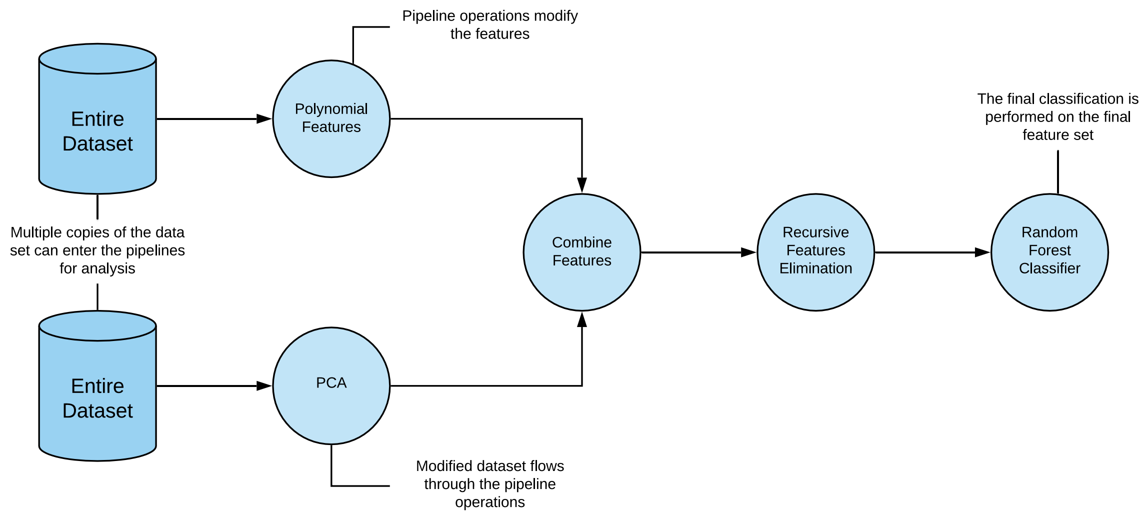
Different industries and organizations have to deal with lot of data, be it related to the products or the employee and this data can be very useful for understanding things properly but every organization can't afford a data scientist.

3.2 TPOT Classification & Regression

```
class tpot.TPOTClassifier(generations=100, population_size=100,
                          offspring_size=None, mutation_rate=0.9,
                          crossover_rate=0.1,
                          scoring='accuracy', cv=5,
                          subsample=1.0, n_jobs=1,
                          max_time_mins=None, max_eval_time_mins=5,
                          random_state=None, config_dict=None,
                          template=None,
                          warm_start=False,
                          memory=None,
                          use_dask=False,
                          periodic_checkpoint_folder=None,
                          early_stop=None,
                          verbosity=0,
                          disable_update_check=False)
```

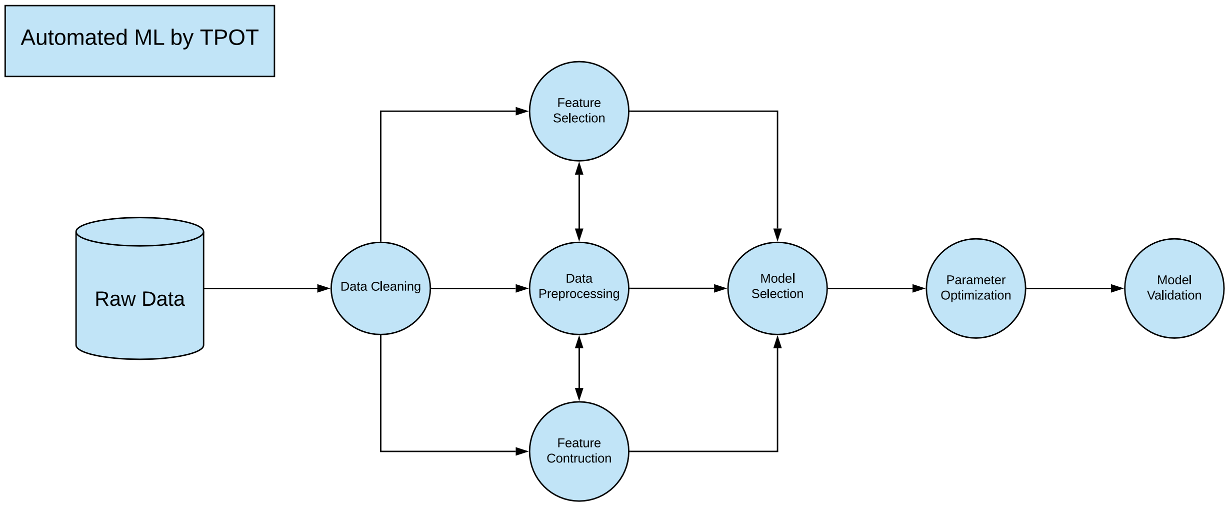
This is the main code that will initiate the TPOT automation process. The code for regression can also be called similarly

How does TPOT Work?


Pipeline operations modify the features
Entire Dataset
Polynomial Features
The final classification is performed on the final feature set
Multiple copies of the data set can enter the pipelines for analysis
Combine Features
Recursive Features Elimination
Random Forest Classifier
Entire Dataset
PCA
Modified dataset flows through the pipeline operations

The **TPOTClassifier / TPOTRegressor** performs an intelligent search over machine learning pipelines which can include supervised models, pre-processors, feature selection techniques, and any other estimators or transformers that follows the sklearn APIs. The TPOT Classifier and Regressor will also search over all the hyper-parameters of all objects inside the pipeline to generate the optimized one.

With default settings, TPOT Classifier/Regressor will search over a wide range of supervised classification algorithms, transformers, and their parameters. However, the algorithms, the transformers, and the hyper-parameters that will be searched over can be fully customized using other parameters.



This is a diagram to show the automation done through TPOT

3.3 AutoViz

AutoViz is the solution to many of the aforementioned challenges that can arise when performing data visualization or solving a dataset. The tool can be called by a single line of code by feeding I with either a pandas dataframe or a csv file.

If the number of observations is very large, AutoViz will automatically take a random sample; like, AutoViz can find the most important features and plot impactful visualizations only using those automatically selected features.

```
from autoviz.AutoViz_Class import AutoViz_Class

AV = AutoViz_Class()
```

First load a dataset (any CSV or text file) into a Pandas dataframe or with the path to the csv file that need visualization. If it doesn’t have a filename (the filename argument), the filename parameter can be assigned as “” (empty string).
This will call the AutoViz class constructer. Now, AutoViz will do everything for us. All the possible charts and plots can be seen on the screen.

The user can also set the sample number of rows and the maximum number of features to visualize by simply passing a parameter to **AutoViz**. AutoViz is also capable of adapting to any number of different data contexts such as regression, classification, or even time-series data. It also delivers output incredibly quickly.

```
filename = ""
sep = ","
dft = AV.AutoViz(
    filename,
    sep,
    target,
    df,
    header=0,
    verbose=0,
    lowess=False,
    chart_format="svg",
    max_rows_analyzed=150000,
    max_cols_analyzed=30,
)
```

This is the main calling program in AV that was used in this project to generate the plots. This will call all the load, display and save programs that are currently outside AV. This program will draw scatter, violin, heatmap and other possible plots for the input dataset and then call the correct variable name with the **add_plots** function and send in the chart created by that plotting program. It must be made sure that **add_plots** function has the exact name of the variable that was defined in the Class AV. Otherwise, it will give an error.

3.4 Categorical Data Encoding

For now, we only support categorical data encoding through **LabelEncoder**. This is due to the fact that the other encoders like **OneHot** or **MultiLabel Binarizer** increases the dimensions of the data, which results poor running speed of the TPOT automation engine. So, while uploading test data, it must be taken care that unseen labels/categories are not present in the dataset, otherwise it will throw an error, and the user will not be able to process the results.

4. Project Details

4.1 Problem Statement

We are trying to make a very simple AutoML Web-App that allows non-experts to make use of machine learning models and techniques without requiring to become an expert in that field. The following points summaries what we are basically trying to achieve:

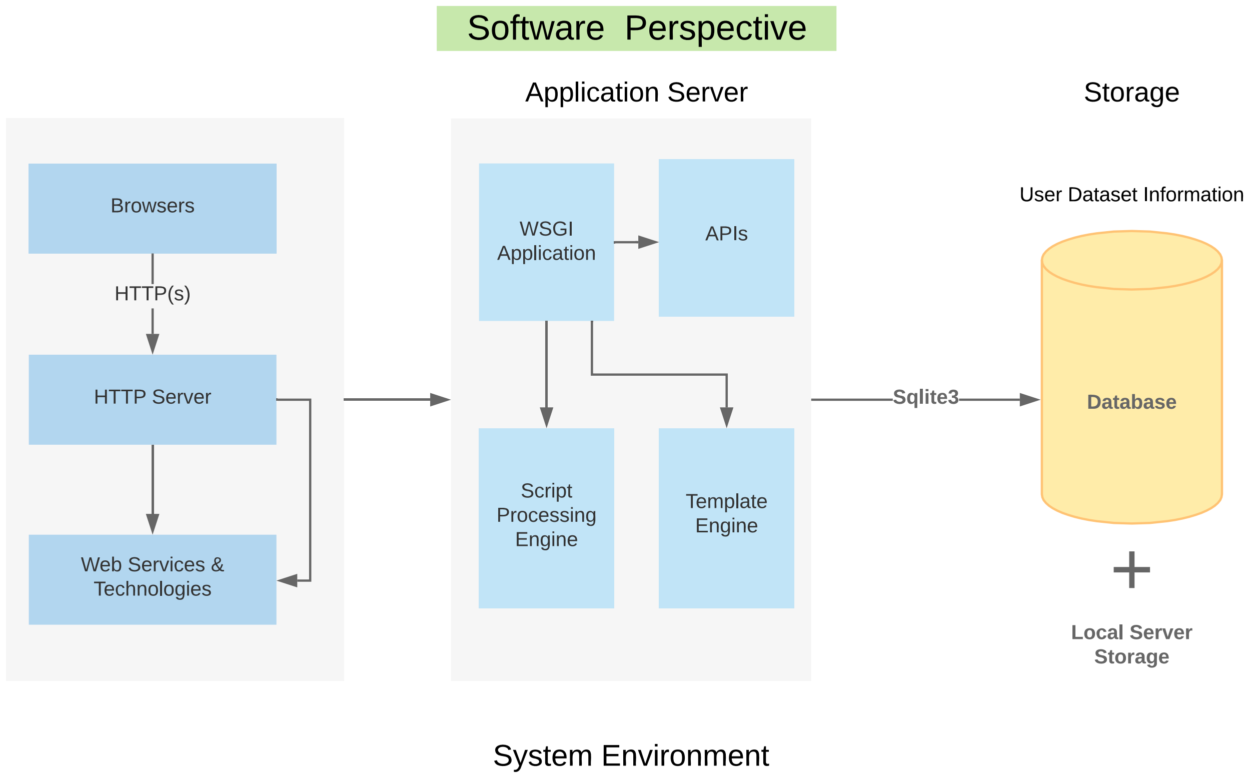
- ✓ Automated data preparation and ingestion from .csv format
- ✓ Automated column type detection: i.e., boolean, discrete numerical, continuous numerical, text, etc
- ✓ Automated feature detection: i.e., label, numerical feature, text feature etc
- ✓ Automated task detection: i.e., binary classification, regression, clustering, or ranking
- ✓ Automated discarding unnecessary columns
- ✓ Automated Hyperparameter optimization
- ✓ Automated analysis of obtained results
- ✓ Manual Model selection; classification, regression or visualization
- ✓ Manual Target Attribute selection
- ✓ User interfaces and visualizations for automated machine learning
- ✓ Logging of all the processed results for further optimization

4.2 Software Functions

There are several major functions that **UnFold It** will perform. Once the user has started the application/website, the software connects to our server. Once a connection has been established, all major functions of the software can be performed. **UnFold It** automatically checks the repository for any changes, and updates all local information accordingly. And the results that comes out of the user's operations are completely based on the corrective-ness of the chosen options by user. If the user feels unsatisfied by the results, he can mail us for any issues.

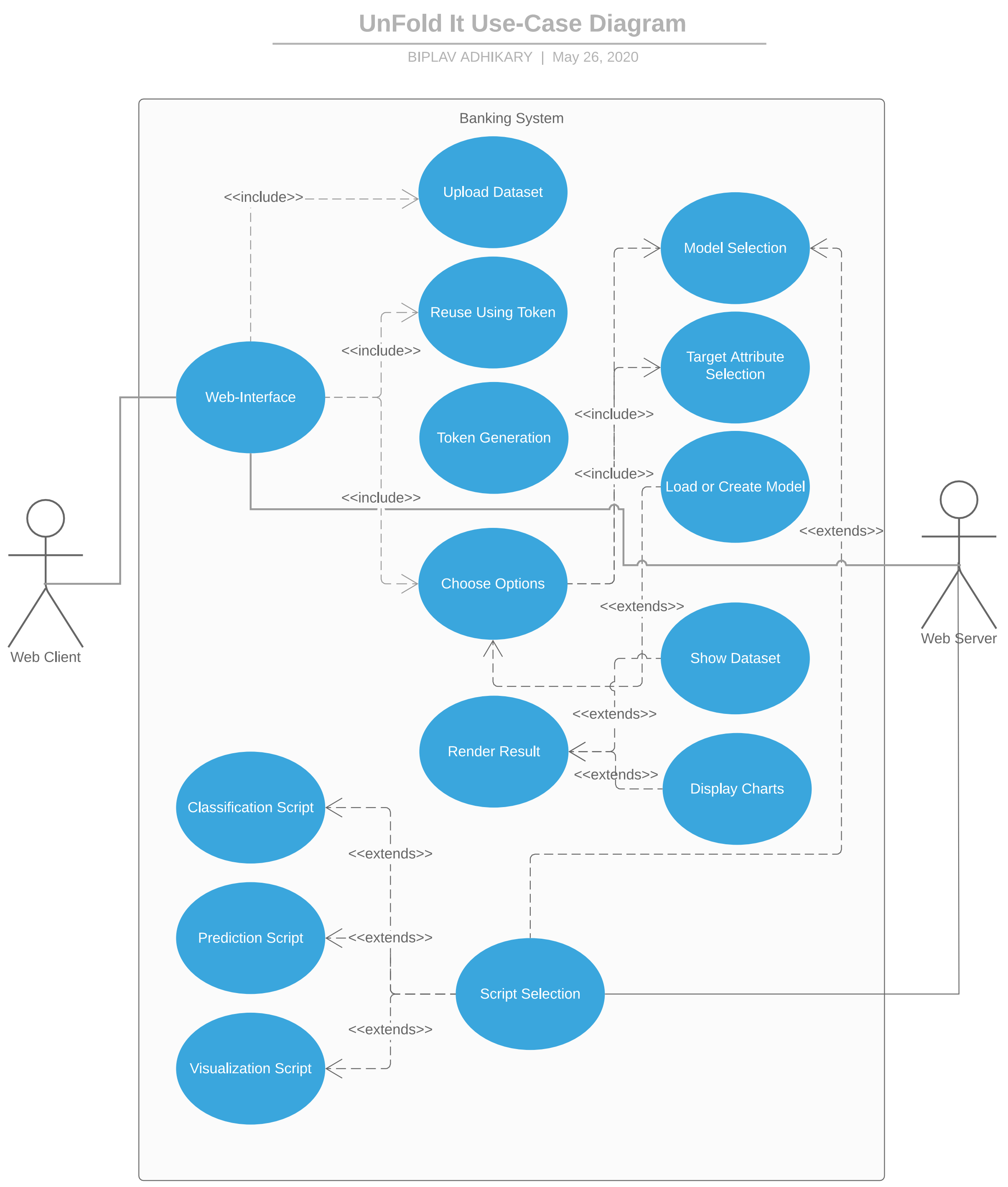
4.3 Software Perspective

The diagram below shows the abstract view of the entire web-application connecting all the major technologies:



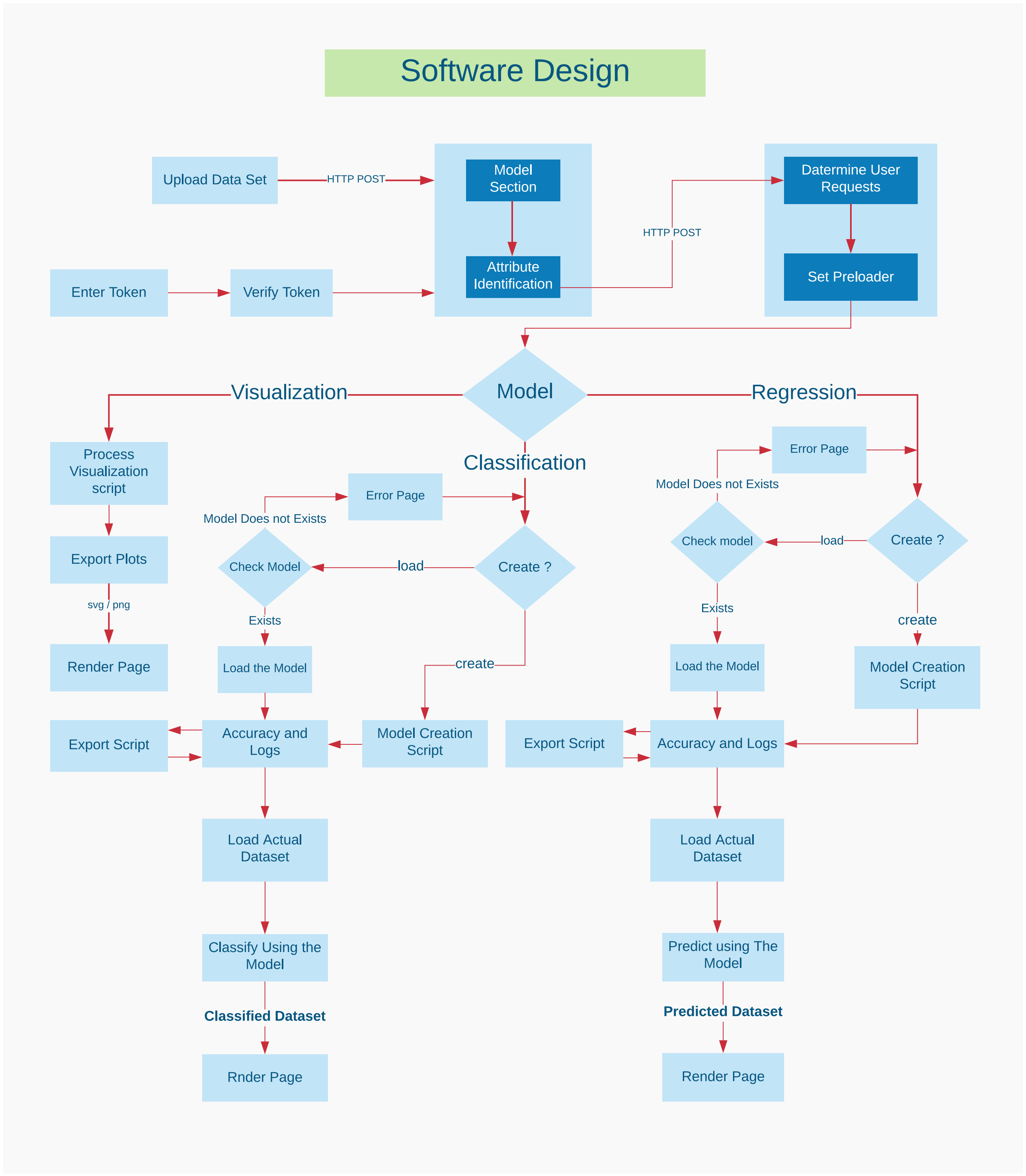
4.4 Basic Use-Cases

The diagram below shows the basic use-case cases of our application:



4.5 Software Design

The diagram below shows the flow control from one component to another



4.6 Project Planning

4.6.1 Methodology

I. Initially, we have done our thorough research in a quest to find the best possible way to implement our AutoML **UnFold_It** Web-app by trying out different combinations of frameworks options that we had.

II. And we have also gone through an ample number of available python libraries to find the ones fitting best in each of the use-case scenarios and we concluded with AutoViz, H20 and TPOT to fit our case.

III. After that we had to understand how we will be implementing the backend scripts for three main problems viz, Classification, Visualization & Prediction and integrate the work as a whole in our web-app.

IV. Once we had a deep understanding of the programming stuff, our main aim was to have a good efficiency in the choice of algorithm that will take minimum time to run.

V. And we collaboratively reached to a decision to restrict the size of the dataset to max of 5Mb for now, and decided that once all the crucial part will be over, we'll start optimizing its functionalities more for improved performance.

VI. During the development of the project we had encountered many problems like the concurrency issue, i.e., different users working on the same dataset at the same time, can clash and give unexpected result, so we had to encode the filenames to a unique one.

VII. Since we are limited by time and performance, so we couldn't invest time in cleaning up unprocessed data, so we had to make tough choice of allowing only pre-processed data. So, the user has to make sure that the data they are entering must be pre-processed one beforehand.

4.7 Types of Accepted Dataset

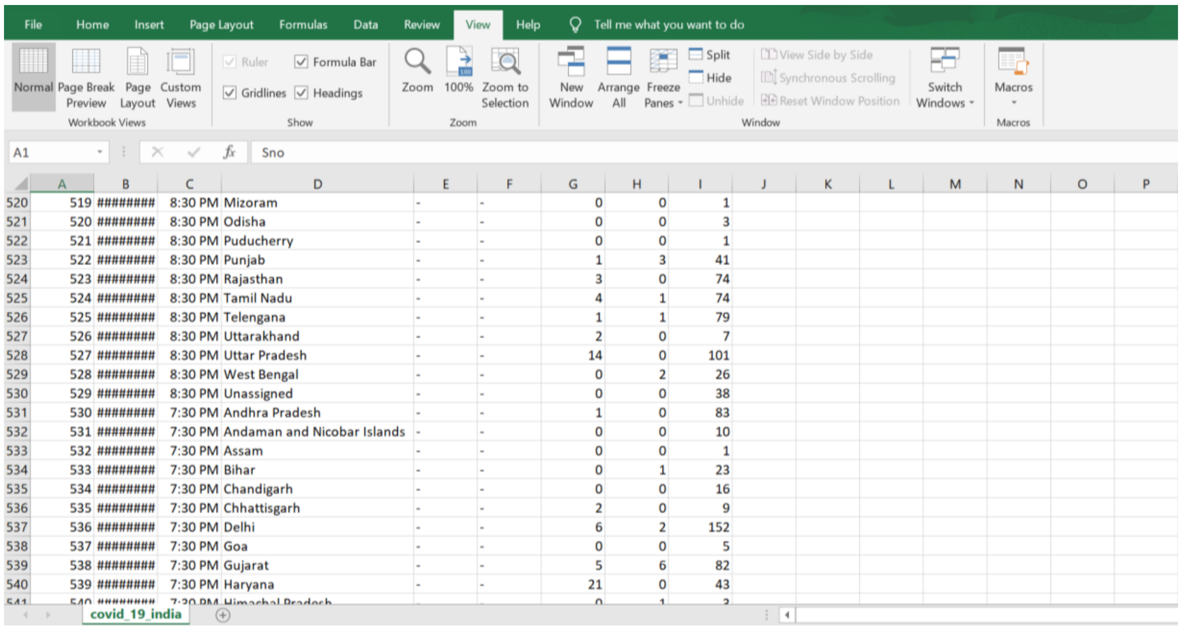
Having the correct dataset is the most important requirement of our project to move ahead in the goal part of the process, and by the correct dataset means it should be Pre-Processed data i.e., the data should be cleaned and accurate to get the desired output. Since, this is the first version of our project we have made some restrictions to the dataset and in its uploading processes for allowing the app to execute smoothly. They are:

- The dataset should be Pre-processed.
- The dataset should be of maximum 5mb not more than that.
- The dataset file should be of type .csv or .tsv.
- If the size of the dataset is greater than that, then it might take longer to process depending on the server limitations and capabilities.

SCREENSHOTS OF AN EXAMPLE DATSET EXPLAINING THE NECESSARY FEATURES

Covid-19-India.csv

It contains all the information about the Covid-19 cases in different places of India.



A1 | Sno

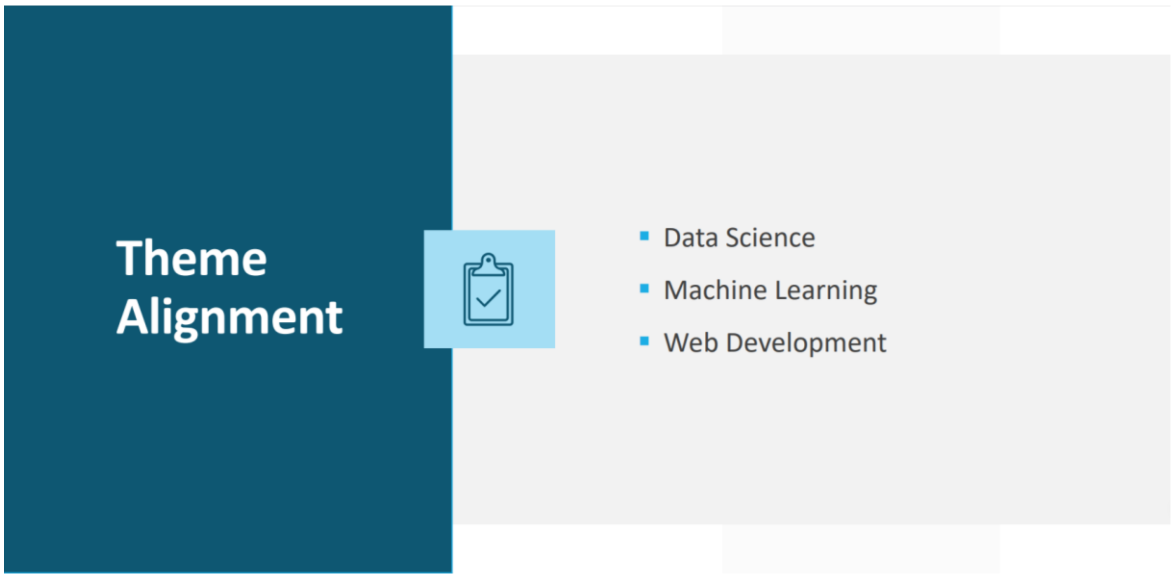
	A	B	C	D	E	F	G	H	I
520	519	########	8:30 PM	Mizoram	-	-	0	0	1
521	520	########	8:30 PM	Odisha	-	-	0	0	3
522	521	########	8:30 PM	Puducherry	-	-	0	0	1
523	522	########	8:30 PM	Punjab	-	-	1	3	41
524	523	########	8:30 PM	Rajasthan	-	-	3	0	74
525	524	########	8:30 PM	Tamil Nadu	-	-	4	1	74
526	525	########	8:30 PM	Telengana	-	-	1	1	79
527	526	########	8:30 PM	Uttarakhand	-	-	2	0	7
528	527	########	8:30 PM	Uttar Pradesh	-	-	14	0	101
529	528	########	8:30 PM	West Bengal	-	-	0	2	26
530	529	########	8:30 PM	Unassigned	-	-	0	0	38
531	530	########	7:30 PM	Andhra Pradesh	-	-	1	0	83
532	531	########	7:30 PM	Andaman and Nicobar Islands	-	-	0	0	10
533	532	########	7:30 PM	Assam	-	-	0	0	1
534	533	########	7:30 PM	Bihar	-	-	0	1	23
535	534	########	7:30 PM	Chandigarh	-	-	0	0	16
536	535	########	7:30 PM	Chhattisgarh	-	-	2	0	9
537	536	########	7:30 PM	Delhi	-	-	6	2	152
538	537	########	7:30 PM	Goa	-	-	0	0	5
539	538	########	7:30 PM	Gujarat	-	-	5	6	82
540	539	########	7:30 PM	Haryana	-	-	21	0	43
541	540	########	7:30 PM	Himachal Pradesh	-	-	0	1	[illegible]

Attribute	Description
Date	The date column tells about the last updated info till that date
Time	The time column tells the last time when the dataset gets updated
State/Union territory	State column contains the name of the state
Confirmed	It contains the value of the confirmed cases of the respective place.
Cured	It tells about the cured patient from the covid-19 disease
Deaths	The total no. of the people has passed away due to this disease

The above displayed table and screenshot image is view of the dataset **Covid-19.csv** briefing about the situation in India with different rows displaying the current scenario of the different parts of India. How many people got infected by the disease? How many of them have been cured? And how many have passed away as a result this disease. These are all shown in the dataset. We can input such a dataset into our web-app and expect it to run smoothly to give the most accurate results. This dataset is a good example of visualization problem. We can input such a dataset into our web-app, and generate the plots from there. We will be showing you the way of *"how to do that"* in the Implementation section.

5. Technologies

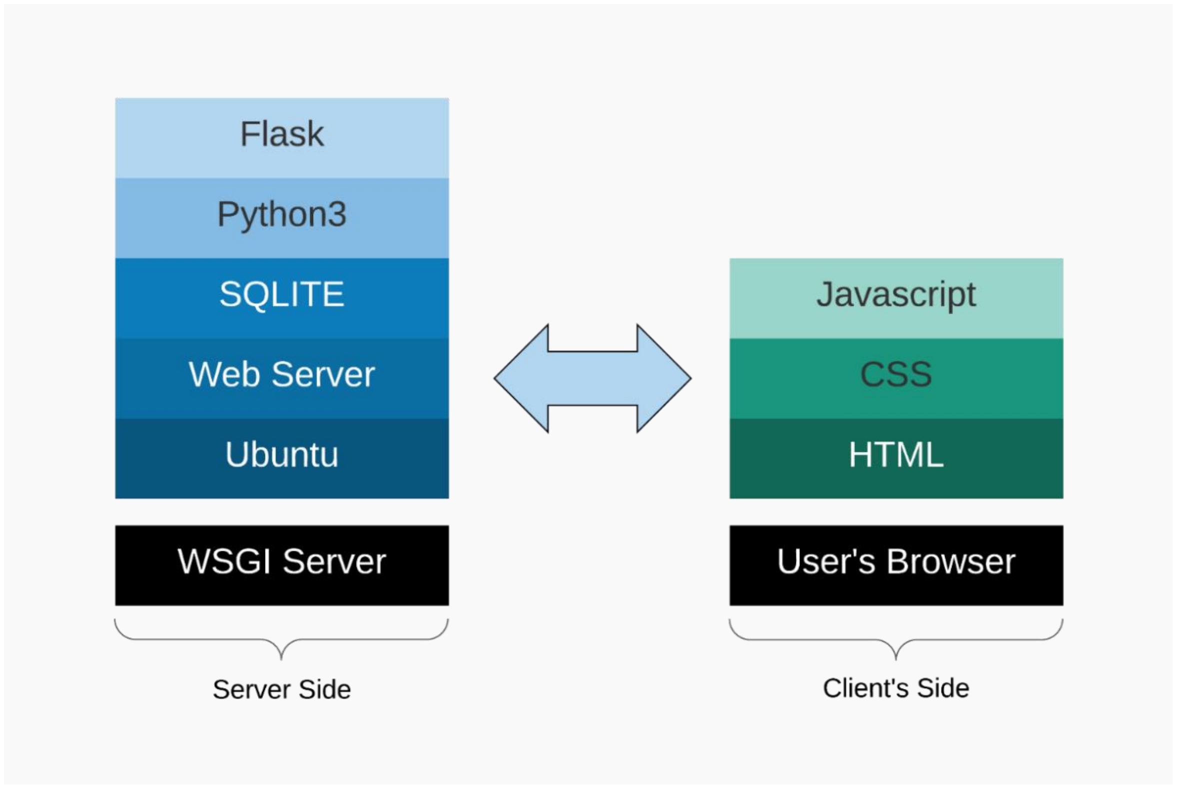
5.1 Theme Alignment



Theme Alignment

- Data Science
- Machine Learning
- Web Development

5.2 Tech Stack



5.3 List of Technologies Used

- Python3

- Machine Learning Libraries:
  - numpy, pandas
  - sklearn
  - automl: tpot, autoviz
  - various other libraries

- Front-End
  - Frameworks:
    - Bootstrap
    - jQuery
  - Vanilla JavaScript

- Backed-End:
  - Flask Web Framework
  - SQLITE3
  - Jinja2 Template Engine

- OS:
  - Linux Ubuntu (Server)
  - Any OS (Client)

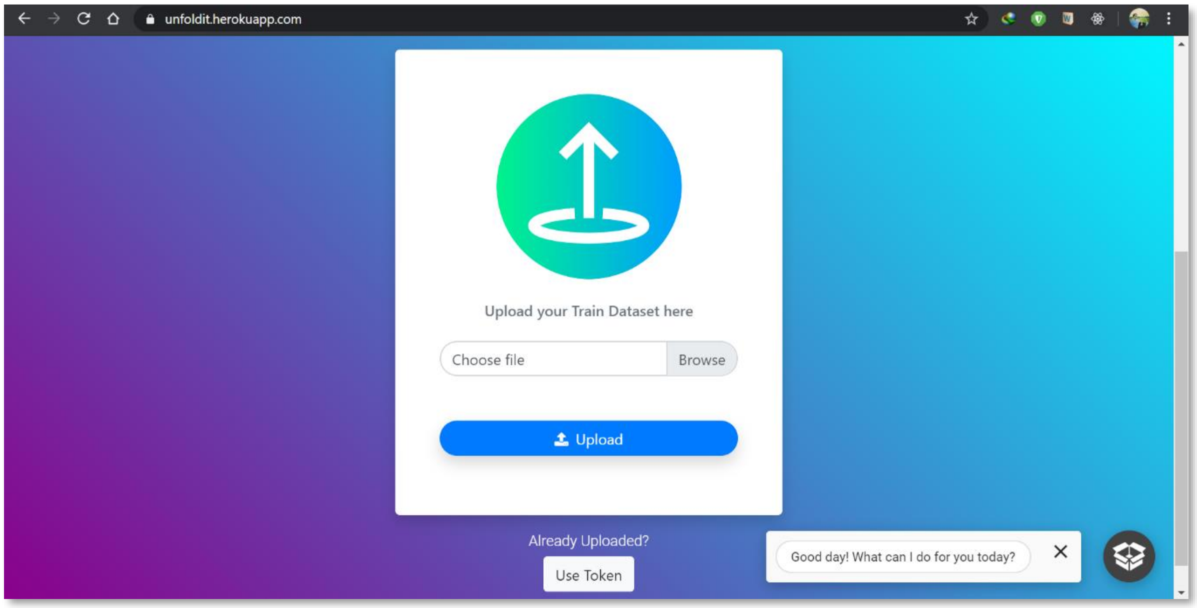
- Storage:
  - Local Storage (Server)
  - SQLITE Database (Server)
  - Browsing Session (Client)

- Hosting:
  - Gunicorn (WSGI Server)
  - Cloud Platform (Heroku)

6. Implementation & Usage

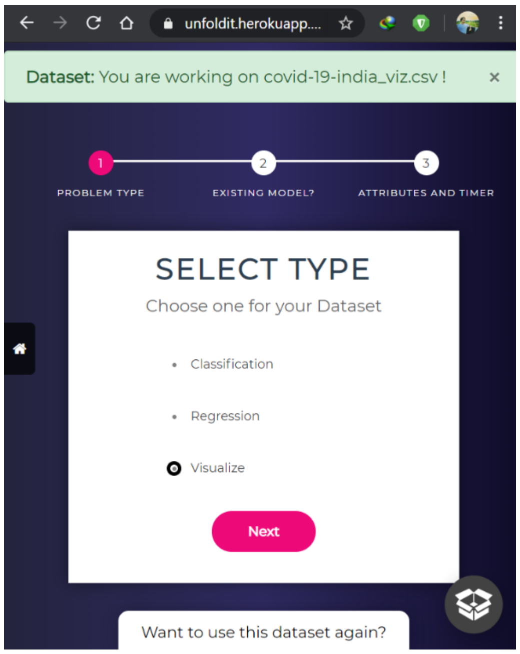
Here are the partial implementations for a few of the features and functionalities of the project. We have not shown everything here, only parts of the major implementations:

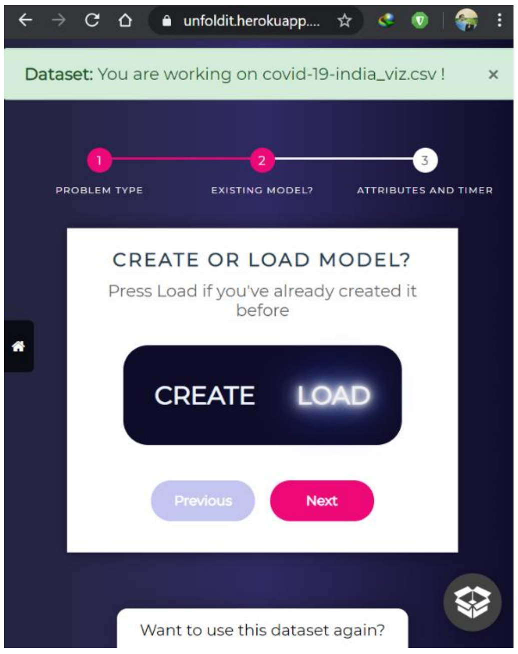
6.1 Data Upload Page



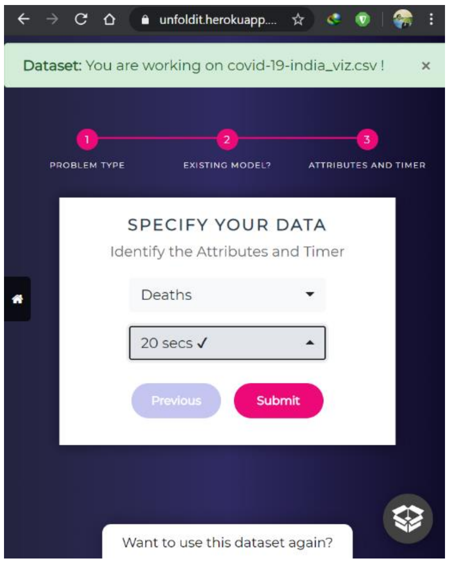
Window where user will get the facility to upload his dataset [preprocessed csv/tsv files].

6.2 Model Selection Window




unfoldit.herokuapp....
Dataset: You are working on covid-19-india_viz.csv !
PROBLEM TYPE
EXISTING MODEL?
ATTRIBUTES AND TIMER
CREATE OR LOAD MODEL?
Press Load if you've already created it before
CREATE
LOAD
Previous
Next
Want to use this dataset again?

User has to choose his problem from the three given types, he will have the facility to generate a token from the bottom of the screen for his uploaded dataset, which he can use to resume his work on the uploaded data.



unfoldit.herokuapp....

Dataset: You are working on covid-19-india_viz.csv !

1 PROBLEM TYPE — 2 EXISTING MODEL? — 3 ATTRIBUTES AND TIMER

SPECIFY YOUR DATA

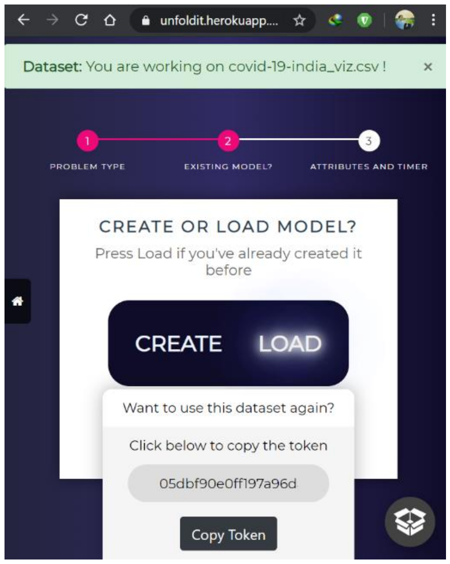
Identify the Attributes and Timer

Deaths

20 secs ✓

Previous | Submit

Want to use this dataset again?



unfoldit.herokuapp....

Dataset: You are working on covid-19-india_viz.csv !

1 PROBLEM TYPE — 2 EXISTING MODEL? — 3 ATTRIBUTES AND TIMER

CREATE OR LOAD MODEL?

Press Load if you've already created it before

CREATE | LOAD

Want to use this dataset again?

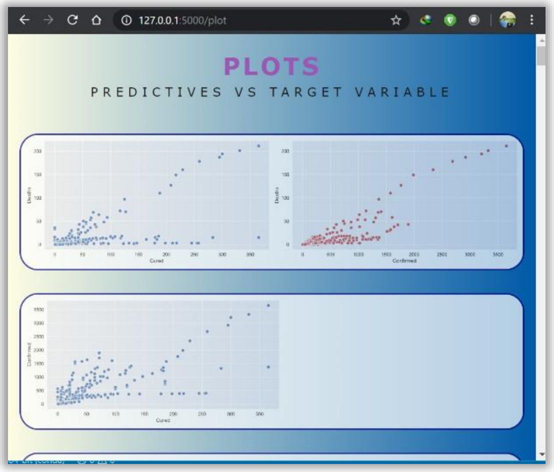
Click below to copy the token

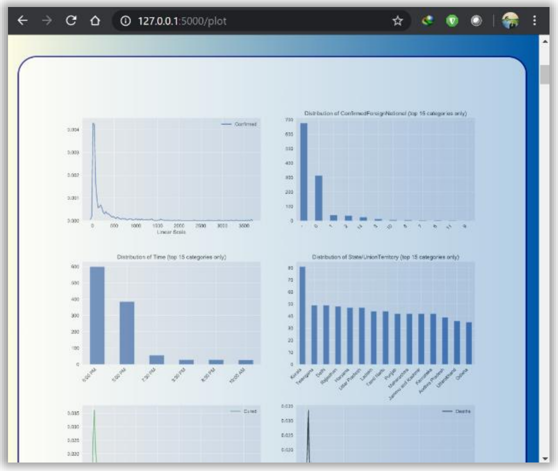
05dbf90e0ff197a96d

Copy Token

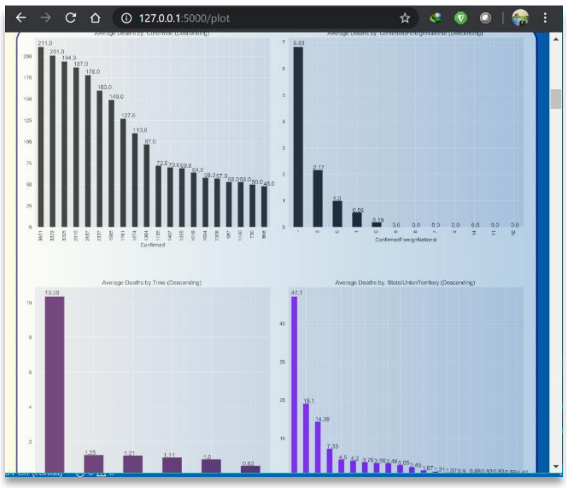
User will have to identify the predictive attribute and choose whether to load a previously created model now or use an existing model generated by our site.

6.3 Visualization (Plot Generation and Display Window)

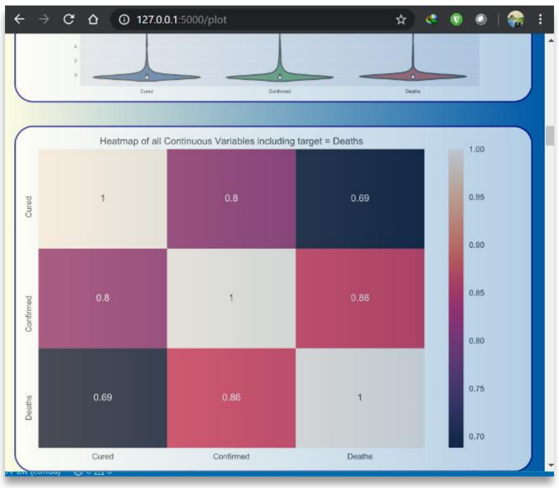




127.0.0.1:5000/plot
Distribution of ConfirmedForeignNational (top 15 categories only)
Distribution of Time (top 15 categories only)
Distribution of State/UnionTerritory (top 15 categories only)
Confirmed
Cured
Deaths

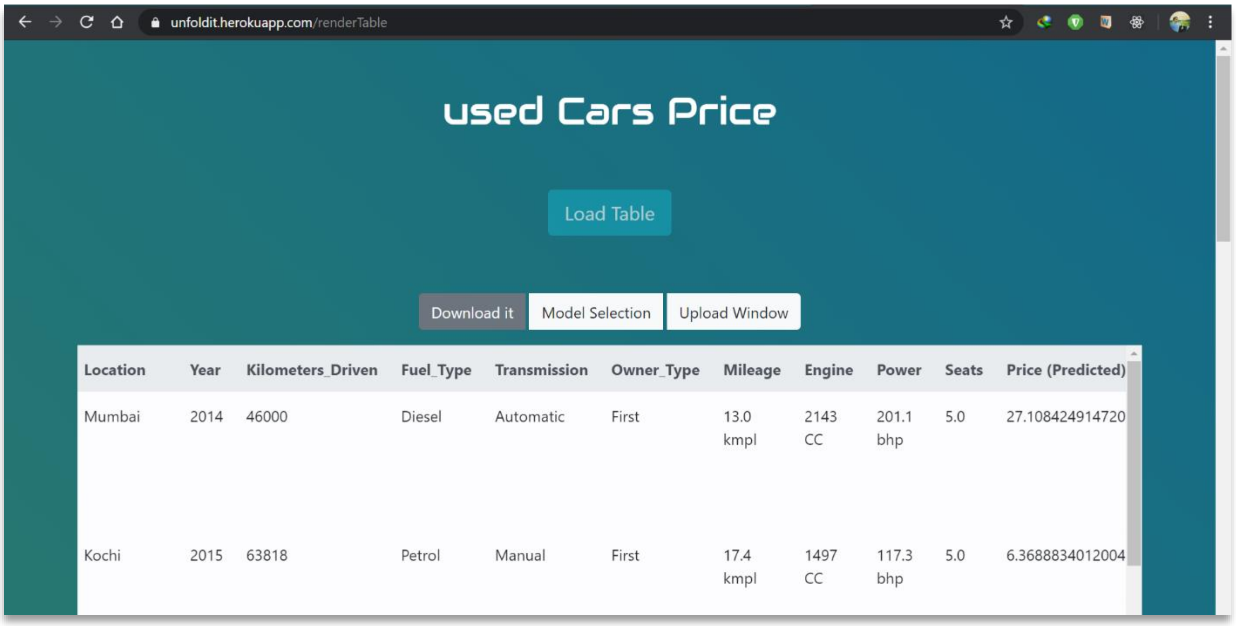


127.0.0.1:5000/plot
Average Deaths by Time (Descending)
Average Deaths by State/UnionTerritory (Descending)
Confirmed
ConfirmedForeignNational



Auto-Visualization Window that displays all possible charts of possible variables vs target attribute

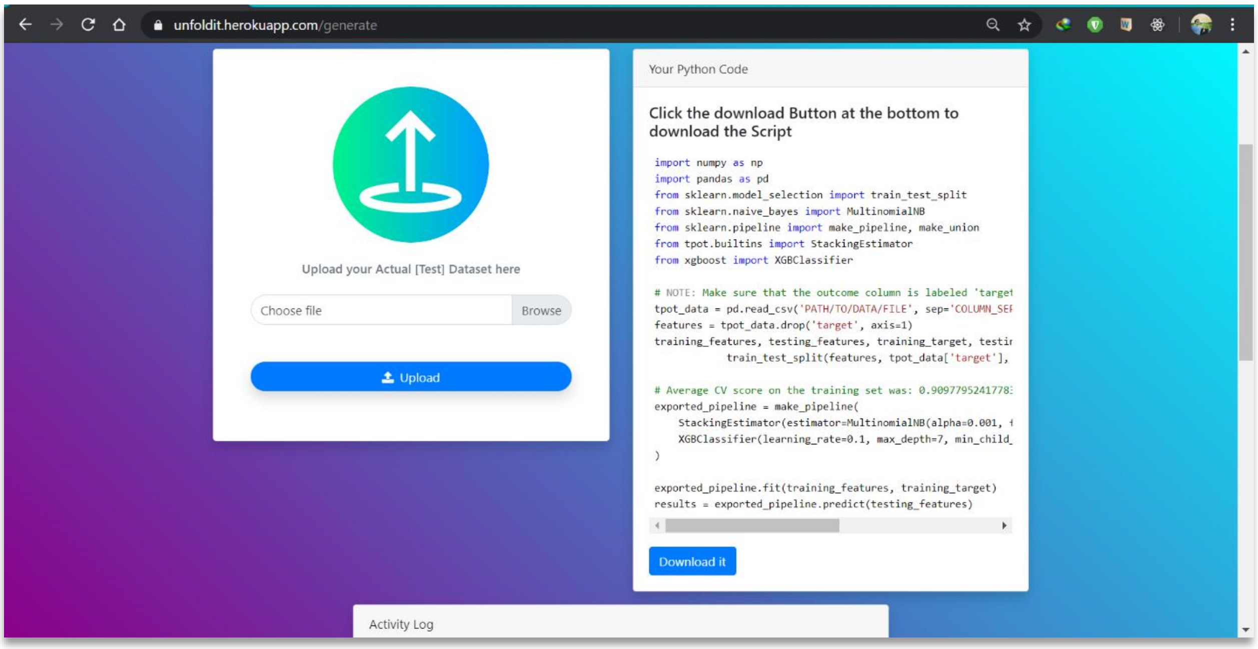
6.4 Predicted Dataset View Window



Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage	Engine	Power	Seats	Price (Predicted)
Mumbai	2014	46000	Diesel	Automatic	First	13.0 kmpl	2143 CC	201.1 bhp	5.0	27.108424914720
Kochi	2015	63818	Petrol	Manual	First	17.4 kmpl	1497 CC	117.3 bhp	5.0	6.3688834012004

Window Displaying the Classified Results of Actual Data from trained Model

6.5 Accuracy and Customized Download Script Window

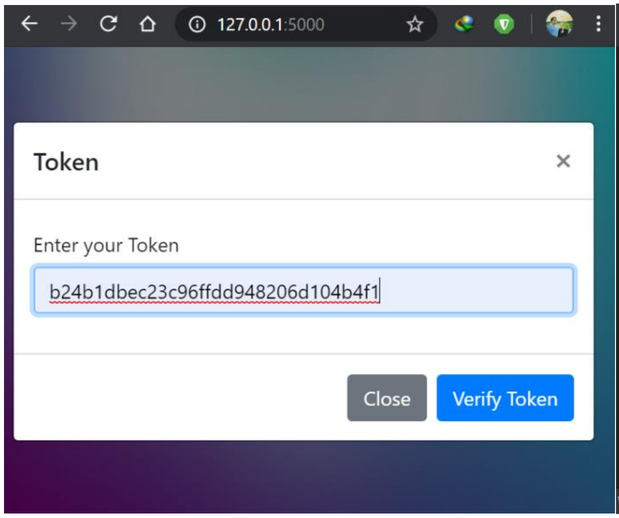

unfoldit.herokuapp.com/generate
Upload your Actual [Test] Dataset here
Choose file
Browse
Upload
Your Python Code
Download it
Activity Log

Click the download Button at the bottom to download the Script

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import make_pipeline, make_union
from tpot.builtins import StackingEstimator
from xgboost import XGBClassifier

# NOTE: Make sure that the outcome column is labeled 'target
tpot_data = pd.read_csv('PATH/TO/DATA/FILE', sep='COLUMN_SE
features = tpot_data.drop('target', axis=1)
training_features, testing_features, training_target, testi
            train_test_split(features, tpot_data['target'],

# Average CV score on the training set was: 0.909779524177
exported_pipeline = make_pipeline(
    StackingEstimator(estimator=MultinomialNB(alpha=0.001,
    XGBClassifier(learning_rate=0.1, max_depth=7, min_child_
)

exported_pipeline.fit(training_features, training_target)
results = exported_pipeline.predict(testing_features)
```

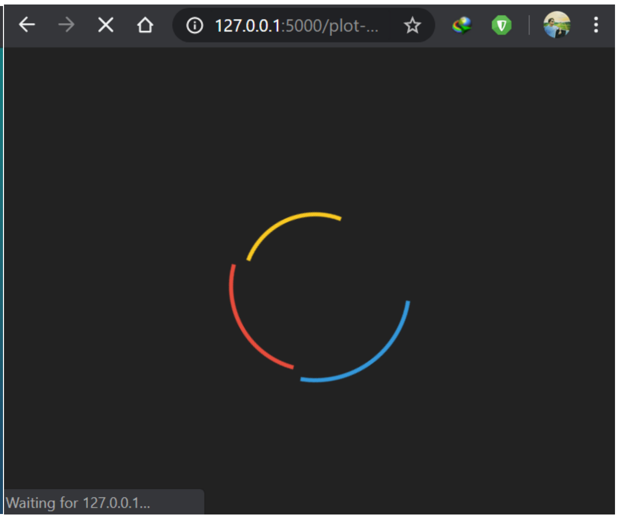
Window Displaying the Classified Results of Actual Data from trained Model

6.6 Miscellaneous Windows



Enter your Token

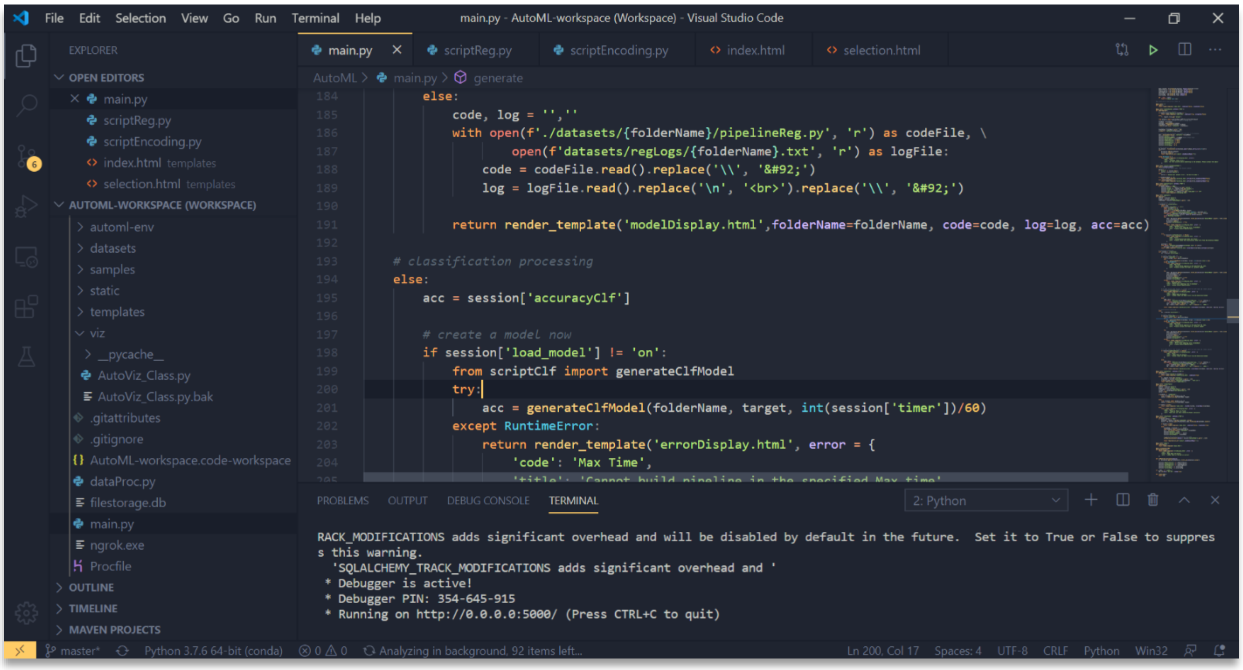
b24b1dbec23c96ffdd948206d104b4f1



Use existing dataset using Token

Loader for Processing

6.7 Project Environment


File Edit Selection View Go Run Terminal Help
main.py - AutoML-workspace (Workspace) - Visual Studio Code
main.py
scriptReg.py
scriptEncoding.py
index.html
selection.html
AutoML > main.py > generate
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
2: Python
master* Python 3.7.6 64-bit (conda) 0 0 Analyzing in background, 92 items left... Ln 200, Col 17 Spaces: 4 UTF-8 CRLF Python Win32

```
            else:
                code, log = '',''
                with open(f'./datasets/{folderName}/pipelineReg.py', 'r') as codeFile, \
                        open(f'datasets/regLogs/{folderName}.txt', 'r') as logFile:
                    code = codeFile.read().replace('\\', '&#92;')
                    log = logFile.read().replace('\n', '<br>').replace('\\', '&#92;')

                return render_template('modelDisplay.html',folderName=folderName, code=code, log=log, acc=acc)

        # classification processing
        else:
            acc = session['accuracyClf']

            # create a model now
            if session['load_model'] != 'on':
                from scriptClf import generateClfModel
                try:
                    acc = generateClfModel(folderName, target, int(session['timer'])/60)
                except RuntimeError:
                    return render_template('errorDisplay.html', error = {
                        'code': 'Max Time',
                        'title': 'Cannot build pipeline in the specified Max time'
```

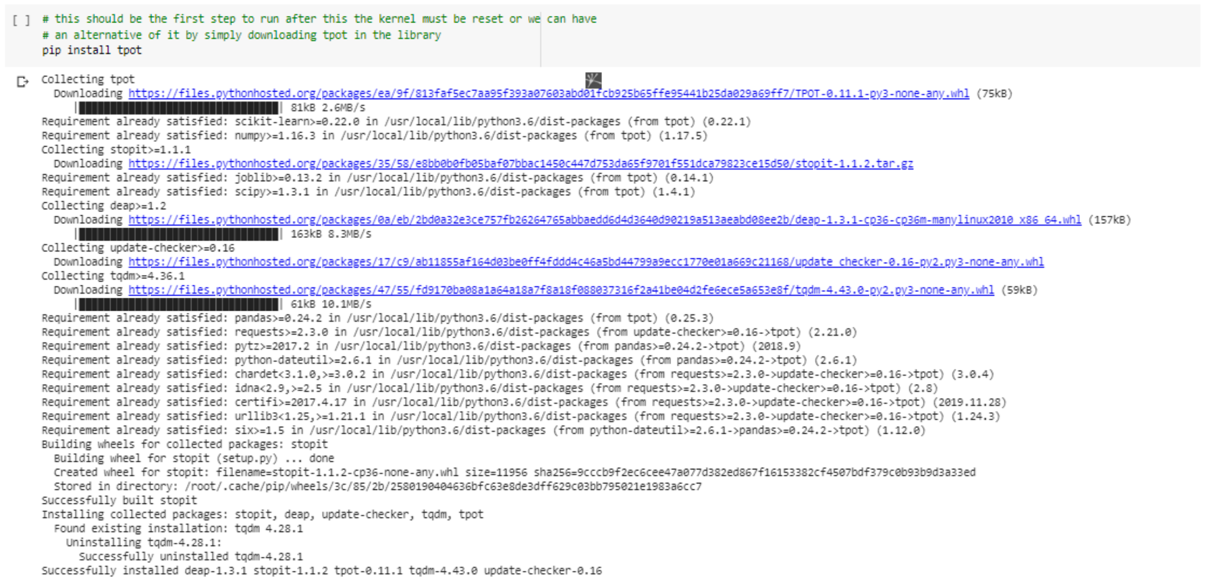
```
RACK_MODIFICATIONS adds significant overhead and will be disabled by default in the future.  Set it to True or False to suppres
s this warning.
  'SQLALCHEMY_TRACK_MODIFICATIONS adds significant overhead and '
 * Debugger is active!
 * Debugger PIN: 354-645-915
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
```

VS Code Window

6.8 Automation Script Implementation (Classification)

Step 1

The first step should be to install the TPOT library after this kernel will be reset. If you are using Jupyter notebook then you need to install it properly

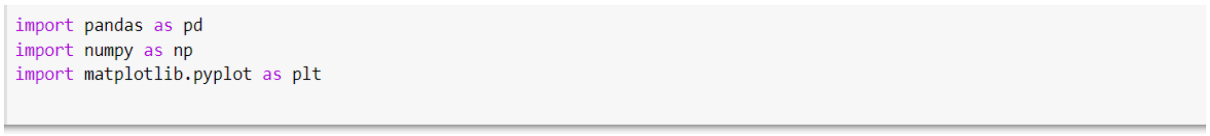


```
[ ] # this should be the first step to run after this the kernel must be reset or we can have
    # an alternative of it by simply downloading tpot in the library
    pip install tpot
```

```
Collecting tpot
  Downloading https://files.pythonhosted.org/packages/ea/9f/813faf5ec7aa95f393a07603abd01fcb925b65ffe95441b25da029a69ff7/TPOT-0.11.1-py3-none-any.whl (75kB)
     |████████████████████████████████| 81kB 2.6MB/s
Requirement already satisfied: scikit-learn>=0.22.0 in /usr/local/lib/python3.6/dist-packages (from tpot) (0.22.1)
Requirement already satisfied: numpy>=1.16.3 in /usr/local/lib/python3.6/dist-packages (from tpot) (1.17.5)
Collecting stopit>=1.1.1
  Downloading https://files.pythonhosted.org/packages/35/58/e8bb0b0fb05baf07bbac1450c447d753da65f9701f551dca79823ce15d50/stopit-1.1.2.tar.gz
Requirement already satisfied: joblib>=0.13.2 in /usr/local/lib/python3.6/dist-packages (from tpot) (0.14.1)
Requirement already satisfied: scipy>=1.3.1 in /usr/local/lib/python3.6/dist-packages (from tpot) (1.4.1)
Collecting deap>=1.2
  Downloading https://files.pythonhosted.org/packages/0a/eb/2bd0a32e3ce757fb26264765abbaedd6d4d3640d90219a513aeabd08ee2b/deap-1.3.1-cp36-cp36m-manylinux2010_x86_64.whl (157kB)
     |████████████████████████████████| 163kB 8.3MB/s
Collecting update-checker>=0.16
  Downloading https://files.pythonhosted.org/packages/17/c9/ab11855af164d03be0ff4fddd4c46a5bd44799a9ecc1770e01a669c21168/update_checker-0.16-py2.py3-none-any.whl
Collecting tqdm>=4.36.1
  Downloading https://files.pythonhosted.org/packages/47/55/fd9170ba08a1a64a18a7f8a18f088037316f2a41be04d2fe6ece5a653e8f/tqdm-4.43.0-py2.py3-none-any.whl (59kB)
     |████████████████████████████████| 61kB 10.1MB/s
Requirement already satisfied: pandas>=0.24.2 in /usr/local/lib/python3.6/dist-packages (from tpot) (0.25.3)
Requirement already satisfied: requests>=2.3.0 in /usr/local/lib/python3.6/dist-packages (from update-checker>=0.16->tpot) (2.21.0)
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.6/dist-packages (from pandas>=0.24.2->tpot) (2018.9)
Requirement already satisfied: python-dateutil>=2.6.1 in /usr/local/lib/python3.6/dist-packages (from pandas>=0.24.2->tpot) (2.6.1)
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in /usr/local/lib/python3.6/dist-packages (from requests>=2.3.0->update-checker>=0.16->tpot) (3.0.4)
Requirement already satisfied: idna<2.9,>=2.5 in /usr/local/lib/python3.6/dist-packages (from requests>=2.3.0->update-checker>=0.16->tpot) (2.8)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.6/dist-packages (from requests>=2.3.0->update-checker>=0.16->tpot) (2019.11.28)
Requirement already satisfied: urllib3<1.25,>=1.21.1 in /usr/local/lib/python3.6/dist-packages (from requests>=2.3.0->update-checker>=0.16->tpot) (1.24.3)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.6/dist-packages (from python-dateutil>=2.6.1->pandas>=0.24.2->tpot) (1.12.0)
Building wheels for collected packages: stopit
  Building wheel for stopit (setup.py) ... done
  Created wheel for stopit: filename=stopit-1.1.2-cp36-none-any.whl size=11956 sha256=9cccb9f2ec6cee47a077d382ed867f16153382cf4507bdf379c0b93b9d3a33ed
  Stored in directory: /root/.cache/pip/wheels/3c/85/2b/2580190404636bfc63e8de3dff629c03bb795021e1983a6cc7
Successfully built stopit
Installing collected packages: stopit, deap, update-checker, tqdm, tpot
  Found existing installation: tqdm 4.28.1
    Uninstalling tqdm-4.28.1:
      Successfully uninstalled tqdm-4.28.1
Successfully installed deap-1.3.1 stopit-1.1.2 tpot-0.11.1 tqdm-4.43.0 update-checker-0.16
```

Step 2

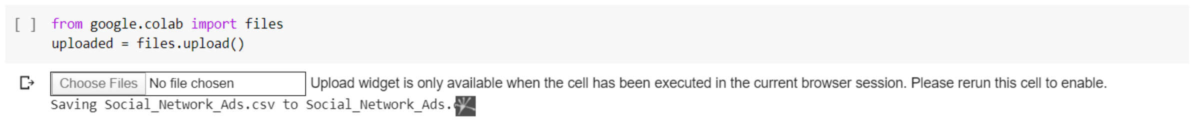
Import the basic library required like pandas, numpy and matplotlib.



```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Step 3

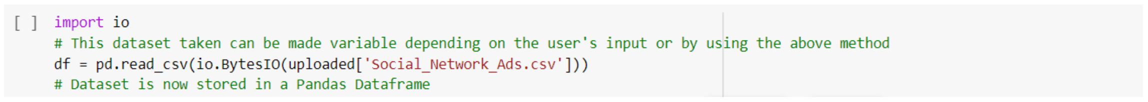
We are using **colab** for this so we will use **google.colab library** for taking the dataset. Here we have taken Social_Network_Ads.csv dataset.



```
[ ] from google.colab import files
    uploaded = files.upload()
```

```
Choose Files  No file chosen  Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving Social_Network_Ads.csv to Social_Network_Ads.
```

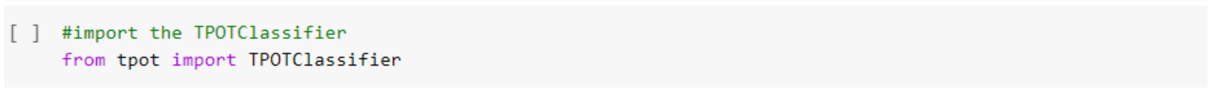
This step involves converting the dataset into pandas dataframe.



```
[ ] import io
    # This dataset taken can be made variable depending on the user's input or by using the above method
    df = pd.read_csv(io.BytesIO(uploaded['Social_Network_Ads.csv']))
    # Dataset is now stored in a Pandas Dataframe
```

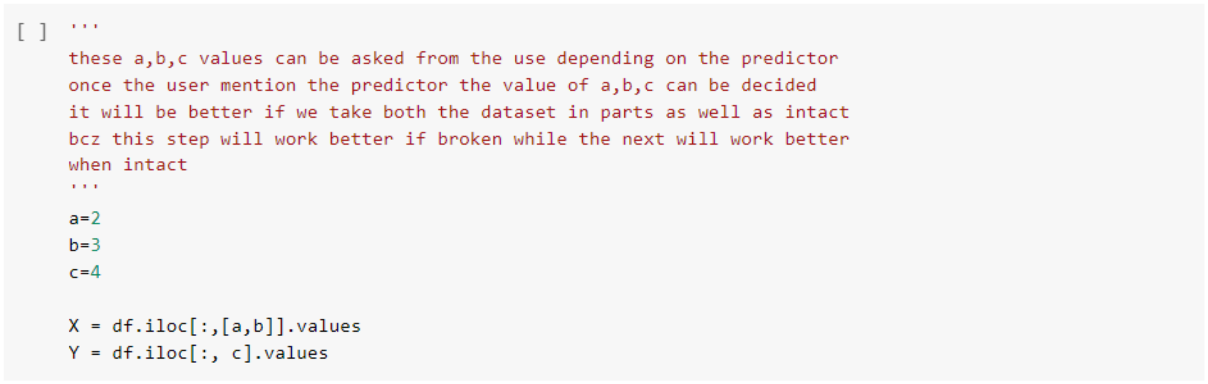
Step 4

Since we are using Tpot library for our ML Automation we need to install TPOTClassifier



```
[ ] #import the TPOTClassifier
    from tpot import TPOTClassifier
```

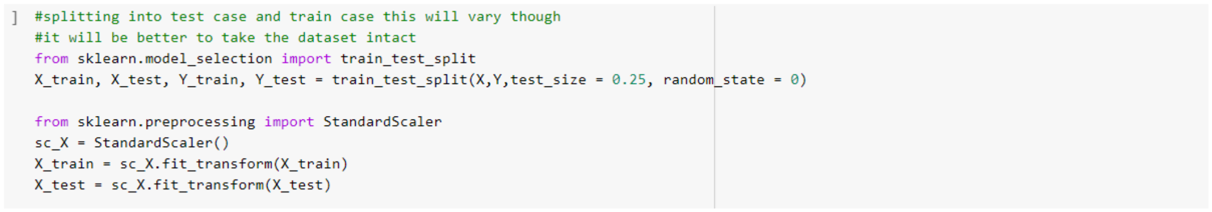
Here the dataset is split into X and Y means the values used for prediction and the values to be predicted respectively.
The values of a, b and c with be decided on the basis of the options selected.
[Note: We will receive the target attribute from the user. For demonstration purpose only, here we are using a random attribute]



```
[ ] '''
    these a,b,c values can be asked from the use depending on the predictor
    once the user mention the predictor the value of a,b,c can be decided
    it will be better if we take both the dataset in parts as well as intact
    bcz this step will work better if broken while the next will work better
    when intact
    '''
    a=2
    b=3
    c=4

    X = df.iloc[:,[a,b]].values
    Y = df.iloc[:, c].values
```

Step 5

Here the X and Y are split into training and testing parts. Training part will be used for making the model and the test part with be used to find the accuracy of the model later on.
Before using the TPOTClassifier we need to **fit_transform** the **X_train** and **X_test** part.



```
] #splitting into test case and train case this will vary though
  #it will be better to take the dataset intact
  from sklearn.model_selection import train_test_split
  X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size = 0.25, random_state = 0)

  from sklearn.preprocessing import StandardScaler
  sc_X = StandardScaler()
  X_train = sc_X.fit_transform(X_train)
  X_test = sc_X.fit_transform(X_test)
```

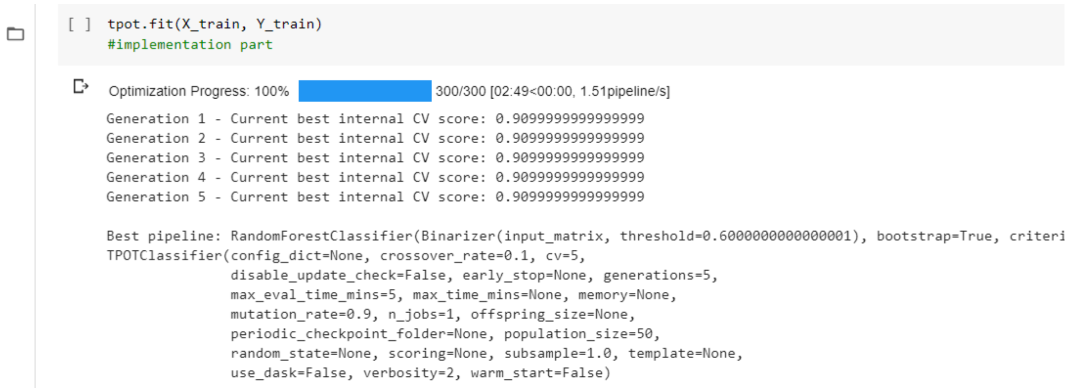
Step 6 (Model Generation)

Here we have created an object of the **TPOTClassifier library** which is named tpot.
We can also set the time for which the model is supposed to run or how many generations will be there.
After that we will fit the object tpot to the **X_train** and **Y_train** to create the model

```
[ ]  tpot = TPOTClassifier(generations=5, population_size=50, verbosity=2)
```

```
[ ]  tpot.fit(X_train, Y_train)
     #implementation part
```

Let’s see how the output comes out for this:
As its visible the progress goes for 2:49 mins and 1.51 pipelines are created every second. Total 300 optimization were done and looking at the CV score it has 90.99% accuracy and **RandomForest** is selected as best classifier.



```
[ ]  tpot.fit(X_train, Y_train)
     #implementation part

Optimization Progress: 100%  300/300 [02:49<00:00, 1.51pipeline/s]
Generation 1 - Current best internal CV score: 0.9099999999999999
Generation 2 - Current best internal CV score: 0.9099999999999999
Generation 3 - Current best internal CV score: 0.9099999999999999
Generation 4 - Current best internal CV score: 0.9099999999999999
Generation 5 - Current best internal CV score: 0.9099999999999999

Best pipeline: RandomForestClassifier(Binarizer(input_matrix, threshold=0.6000000000000001), bootstrap=True, criteri
TPOTClassifier(config_dict=None, crossover_rate=0.1, cv=5,
               disable_update_check=False, early_stop=None, generations=5,
               max_eval_time_mins=5, max_time_mins=None, memory=None,
               mutation_rate=0.9, n_jobs=1, offspring_size=None,
               periodic_checkpoint_folder=None, population_size=50,
               random_state=None, scoring=None, subsample=1.0, template=None,
               use_dask=False, verbosity=2, warm_start=False)
```

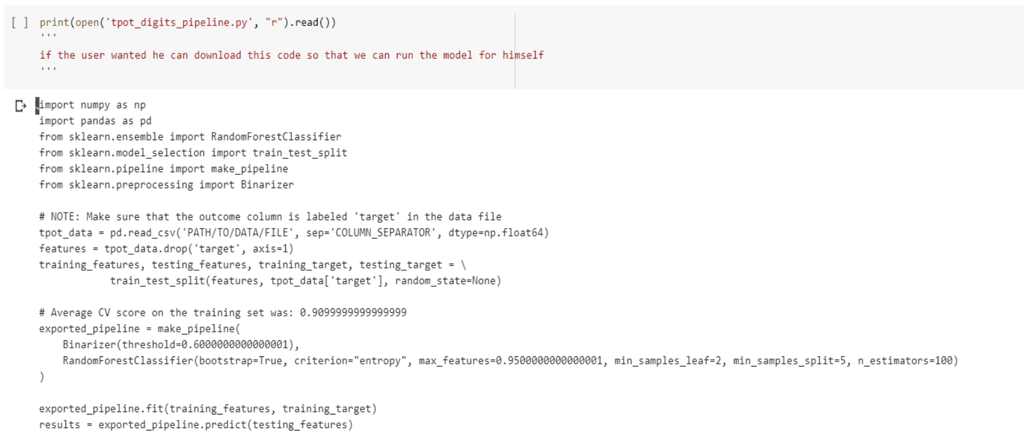
Step 7

This is used to get the accuracy or to know how efficient our model is.

```
b=print(tpot.score(X_test, Y_test))
#calculating the accuracy
```

Step 8

The used can also see the code for the selected pipeline, this way if he wishes he can implement the same code for himself as well.



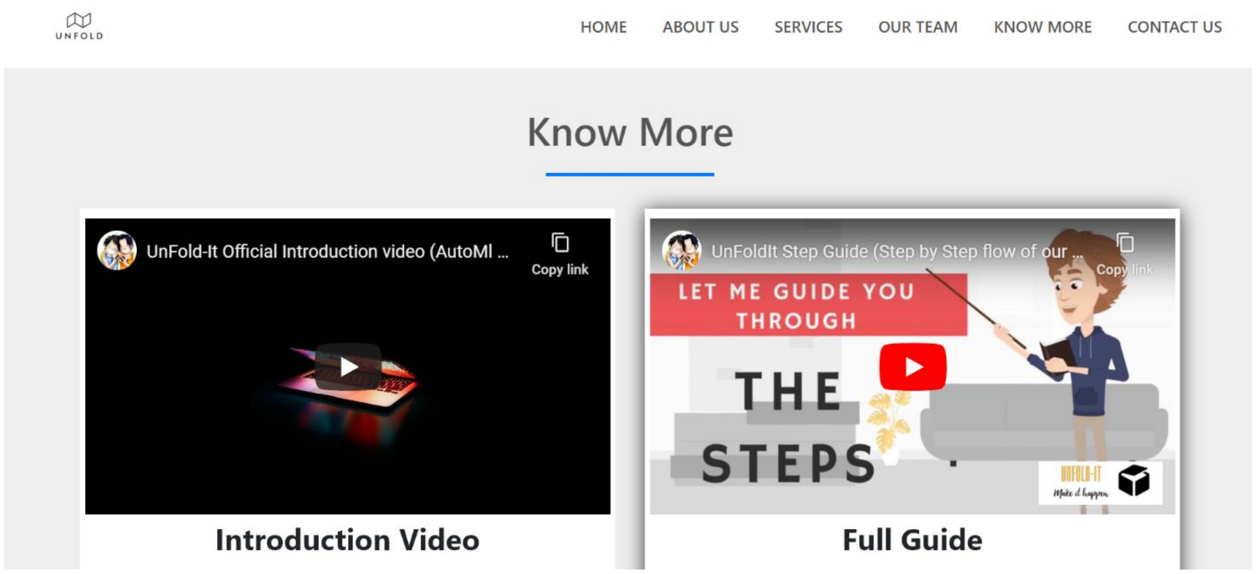
```
[ ]  print(open('tpot_digits_pipeline.py', "r").read())
     '''
     if the user wanted he can download this code so that we can run the model for himself
     '''

import numpy as np
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import Binarizer

# NOTE: Make sure that the outcome column is labeled 'target' in the data file
tpot_data = pd.read_csv('PATH/TO/DATA/FILE', sep='COLUMN_SEPARATOR', dtype=np.float64)
features = tpot_data.drop('target', axis=1)
training_features, testing_features, training_target, testing_target = \
            train_test_split(features, tpot_data['target'], random_state=None)

# Average CV score on the training set was: 0.9099999999999999
exported_pipeline = make_pipeline(
    Binarizer(threshold=0.6000000000000001),
    RandomForestClassifier(bootstrap=True, criterion="entropy", max_features=0.9500000000000001, min_samples_leaf=2, min_samples_split=5, n_estimators=100)
)

exported_pipeline.fit(training_features, training_target)
results = exported_pipeline.predict(testing_features)
```

6.9 Chat-Bot

A chat-bot was also created to the allow the users to put common queries and problems that they might face while using our app. The bot was created using Google DialogFlow's NLP app. The bot has been trained with common phrases and will try its best to give proper response to those queries. Intents and responses were also created in such a way that the bot will be able to handle those common questions and provide a proper response.

6.10 Home Page

This page contains all the basic details about the product, the involved members, technologies and various other tracks.



Here's a snapshot of a section containing the demo and guide videos.

7. Comparison

Although, a manual modeling design approach is emphasized in many statistic articles and textbooks, where the practitioner performs some exploratory analysis to assess the neccesary factors including whether there's any collinearities in the predictors; if any features can be removed from the dataset to reduce the dimensionality; removing outliers; if any variables need transforming; in addition to numerous diagnostics on the error term, such as its variance and distribution.

That's pretty much all fine for exploratory analysis where we're trying to build a model that best explains the data, but if it's predictive easiness which is the ultimate goal, then such a manual approach seems too time consuming, tedious and impractical.

When we develop a model to be used for predicting future events, the key evaluation measure should be how well it can generalize, which is typically achieved through a resampling method such as cross-validation or bootstrapping. Thus, the effect of any transformation or predictor selection step needs to be evaluated on a holdout set, which would be very time consuming if performed manually over 10-fold cross-validation.

For example, if we want to reduce the number of variables, then we would need to perform a step-by-step procedure to select the optimal predictors for each resampling iteration, as the chosen predictors may change depending on the holdout set. Another approach would be to manually choose some subsets of the predictors before fitting any model, so as to make sure we are not inputting any bias into the procedure, and run cross-validation on models trained on each of these.

8. Results & Discussions

8.1 Constraints

- Pre-processed data and clean data sets are required.
- User must have the very basic knowledge that how ML feature works.
- Large datasets might take long time to process.
- User's connection must fast enough for large datasets
- Limited to HTTP/HTTPS. This system is working with single server.
- Useless columns must be removed like ids, date-created, etc., and test categorical data must not have new labels from train.

8.2 Assumptions and Dependencies

- The right predictive attribute needs to be chosen. Otherwise the predicted results might not have actual relations with the data.
- User must know the type of problem he is dealing with.
- The user is having a stable internet connection.

8.3 User Characteristics

There are no major prerequisites required for using our application. The user must be aware of the very basic terms related to Machine Learning, so he can infer meanings from predictions as well as visualizations. He should be well acquainted with the use of web browsers in a computer system or navigating through the app.

8.4 Expectations

- AutoML algorithms aren't intended to run for only a few minutes but we can do so, but doing so that might lower the accuracy.

- AutoML algorithms can consume a long time to finish their search. AutoML algorithms aren't as simple as fitting one model on the dataset. They will be processing through multiple machine learning algorithms (decision tree, random forests, linear models, SVMs, etc.) and generate the best results in a pipeline with multiple pre-processing steps (missing value imputation, scaling, PCA, feature selection, etc.), the hyper parameters for all of the models and pre-processing steps as well as multiple ways to collect or stack the algorithms within the pipeline.

- AutoML algorithms can recommend various solutions for the provided dataset

8.5 Tests and Results

Dataset Name	*Token*	*Problem Type*	*Max Time Allotted*	*R2Score /plots*	*Inference*
SOCIAL_NETWORK_ ADS_CLF.CSV	fa8bbe79ea18ad5616a5955 c2ef905e8	Classification	1 minute	92.5	Good Accuracy
GOT-CHARACTER-PREDICTIONS.CSV	c2179312ffb9c9e3f19cb9fa 3540f1f8	Visualization	No restrictions	49 subplots	Was able to generate all possible subplots
BC_CLF.CSV	8b56ff6e55aa96a7a7c8cdc 1087c6e80	Classification	1 minute	98.245	Very good accuracy. Model doesn't seem to be overfitted
BIKE-BUYER_CLF.CSV	dee011c3c8857b6f2f9df7e 11acb8324	Classification	20 secs	92.619	Giving Good Predictions
BOSTON_REG.CSV	2b44ad82343d2852a0f24ba e110c3d46	Regression	20 secs	66.96	Poor Accuracy due to low time constraint
CARS_CLF.CSV	010d07e56b8128c3e688ad 51a76cd7b6	Classification	15 secs	97.109	Very good accuracy. Model doesn't seem to be overfitted
COVID-19-INDIA_VIZ.CSV	05dbf90e0ff197a96da7a3f9 98c020f9	Visualization	No restrictions	26 subplots	Could generate all the necessary plots
HEART_CLF.CSV	dd80d10a7bb290e8df3ae72 34cbe07fb	Classification	15 secs	77.419	Predictions are close enough for maximum data
INDIAVIX_REG.CSV	8255b5040ffb72e33d671fb 597e3c67f	Regression	1 min	98.791	Looks good.
REAL ESTATE_REG.CSV	9a84aa0059a062531efcb72 db6eb1453	Regression	1 min	72.362	Predictions are OK, but less accuracy due to outliers
CARS PRICE_REG.CSV	a9df313e069c1c6de8babb2 eefd46301	Regression	1 min	82.143	Average Score, decent predictions
VOICE_CLF.CSV	7ff0066cb2e10252f5e2158 063e2427a	Classification	3 mins	97.791	Very Close Predictions
WINEQUALITYREDS_R EG.CSV	e8f958761b8b7c06c84ef4d 1640cb587	Classification	20 secs	70	Average Predictions
HR_VIZ.CSV	8acc0aeab0917d821d00507 d5e854631	Visualization	No restrictions	15 subplots	All kind of possible plots are captured

8.6 Conclusion

Based on the tests, our app **UnfoldIt** was able to serve its purpose by enabling users to make nearly accurate predictions based on the trained data, if it gets sufficient time to train the model. Hence, we can conclude that, it is a **Machine learning Product** that allows not only developers, but anyone with limited Machine Learning knowledge and skillset to train high-quality models and pipelines detailed to their personal or professional needs, subjected to the time frame the process has been allotted.

A. Appendices

The team comprised of four members and each one of them had contributed well on their part. The efficiency of the team is surely reflected in the work showcased above.

Avinash Soni (1706119)

He has done the static part of the website by creating and designing the **entire homepage** with responsiveness, modern looks and design through Bootstrap, custom CSS styles and Javascript to ensure that users will be able to feel the simplicity and fluidity while using it. He also had some role in each and every step of the project as it progressed with time, he was the one to provide the idea as to how to implement the project in terms of documentation, project planning and methodology lying under the work exhibited. The future features planning and selection in the website is also thanks to him. He is the leader of the team whereby he tried his best in managing the team and dividing the work among the members of the group and so anyone can come up with an innovative solution even when any member of the team got stuck with things related to the project. Although, he has not played an active part in ML development but he has kept the basic knowledge of ML, but he has always been an active member in suggesting ideas whenever needed. Being a team leader, he has always kept himself open to ideas and appreciated further new ideas from different sources a

Ayush Agrawal (1706122)

Highlights:

- AutoML Regression and classification script designing
- Project idea
- Project director related task
- Validation and Verification testing with Biplav
- Video Designing and Editing
- Logo and Slogan Designing
- Research work and Data analysis
- Suggesting in designing and project flow

As duties of a project director is responsibility for monitoring build progress, overseeing finance and ensuring project quality also making strategic decisions and providing leadership and direction to the project.

The mind behind the project idea of implementing the AutoML and how to implement. Responsible for understanding the project objective and the advantages people can have by using such a website. Also played a major role in keeping the team encouraged and excited to complete the project with great success. Along with Biplav Adhikary responsible for coming up with ways to implement the website and flow in which the website will proceed. Responsible for handling all the ML related parts from creating the scripts to the documentation part. Implementing the script for AutoML and deciding which library will act best in this scenario, after the selection of **TPOT** library coming up with the best possible ways to implement it. Once the website was ready, testing of various datasets was done along with Biplav Adhikary for analyzing their accuracy. Suggesting Biplav Adhikary at every step which will be a better option in project designing phase. Also responsible for research on TPOT library and other AutoML libraries along with the documentation part for ML related definitions, diagrams and Automation script implementation.

Coming up with project logo design and slogan. Designing a video on the introduction to our project and how to use it, what all people can use it and how it can be beneficial for them.

Biplav Adhikary (1706124)

Highlights

- Server-side coding and web front-end building, design and management
- System design, implementational details and flow of data
- The one to create the visualization script and all the utility functions required from data processing through model building to prediction and plot generation.
- DB administrator
- Project Integration and merging all the individual components as a whole from the all the contributors into one product
- Unit testing for each module
- Validation against multiple datasets to check the accuracy & results with Ayush
- Server Hosting with Gunicorn in Heroku
- Contributor of the official documentation
- Step by Step Demonstration Video for using the App

The sever side coding was done with **Flask Application Framework** with each route pointing to an individual URL, which generates the required templates upon receival of requests and processing of the data for rendering pages. It was designed with each route's response time in mind to make sure that responses are fast that I would not result in a timeout issue, except for the model building phase, which by default needs time to find the best suitable pipeline for the dataset.

For storing the data information, an **SQLITE** database was used with **SQLAlchemy** on top of it, which could store fields like id, name, attributes and details regarding each model. And the most frequently used data in the server are stored in client's device as encrypted sessions-data so that they can be retrieved faster anytime without the needing much of request arguments which can be tedious at times. Similarly, the processed models and the plot data are stored in the server's local storage ie, datasets folder. The folder names for each dataset is assigned a unique hashkey, viz., token so that there won't be any conflict among similar filenames. This token can be collected by user upon uploading the dataset once, so that the user can later check back later and resume the work on the same dataset.

The front-end was implemented mainly through bootstrap for mobile friendliness, compatibility and responsiveness along with other Js libraries like **jQuery**, **HighlightJs**, **Font-awesome**, etc. through fastest CDNs. Customized CSS requirements and Js animations are added to give a smooth feel to the end user. And from a functionality point of view, most the necessary checking and anomalies are handled in the front end itself, like checking input fields, generating alerts, etc.

Also, one of the editors of the document along with the other contributors to provide insightful contents and diagrams to the readers to easy grasp and understandability.

Gulam Muhammad (1706130)

Implemented the project in terms of documentation, writing the research paper. Analysis of the research papers has been done and their outcomes are compared. He's is also partially responsible creating the software design and making neat and fantastic diagrams for this documentation.

He is the one to create the chat-bot through **the Google Dialog Flow** app. He has tried his best to train the bot with all suitable phrases so it would be able to grasp the user's intentions and make proper responses, which would help users of our App to get meaningful information so as to help him overcome any difficulties, whether it is in the Model Building phase, or it is in generating predictions or plots.

B. References

Reference Name	**Link**
Hands-On Machine Learning [Aurellien Geron]	https://www.academia.edu/37010160/Hands-On_Machine_Learning_with_Scikit-Learn_and_TensorFlow
Python	https://docs.python.org/3/
Flask	https://flask.palletsprojects.com/en/
SQLAlchemy	https://docs.sqlalchemy.org/en/13/
BootStrap Documentation	https://getbootstrap.com/docs/4.1/getting-started/introduction/
Jquery Documentation	https://api.jquery.com/
Wikipedia	https://wikipedia.com
HTML, CSS, JavaScript Guide	https://www.w3schools.com/
AutoViz	https://github.com/AutoViML/AutoViz
StakeholderMap	https://www.stakeholdermap.com/
Diagrams [Created using LucidChart Web Application]	https://www.lucidchart.com/
Sklearn	https://scikit-learn.org/stable/user_guide.html

TPOT content	https://towardsdatascience.com
AutoML content	https://epistasislab.github.io/tpot/using/
AutoML Library Details	https://www.automl.org/automl/
Environment [ML Script]	https://colab.research.google.com/notebooks
Web Design Ideas and Animations	https://codepen.io/
Background Combinations	https://uigradients.com/
Common Solutions	https://stackoverflow.com/
YouTube Tutorials	https://youtube.com/