Used Cars Condition Evaluation

# Technical Design Document

Version 1.0

RESTRICTED DISTRIBUTION

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

Document Version Control

|  |  |  |  |
| --- | --- | --- | --- |
| Date Issued | Version | Description | Author |
| 25th August 2020 | 1.0 | Initial Draft | Veda E |

Contributors

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

|  |  |
| --- | --- |
| Name | Section Worked Upon |
| Veda E | Initial Draft |

Document Classification

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

Contents

[1 Technical Design Document 2](#_Toc49724604)

[2 Introduction 4](#_Toc49724605)

[2.1 Problem Statement: 4](#_Toc49724606)

[2.2 Objectives: 4](#_Toc49724607)

[3 Workflow Overall: 5](#_Toc49724608)

[4 Libraries used 6](#_Toc49724609)

[5 Data Preprocessing 7](#_Toc49724610)

[5.1 Import dataset from Database(MongoDB): 7](#_Toc49724611)

[5.2 data insights: 9](#_Toc49724612)

[5.3 Checking missing values: 10](#_Toc49724613)

[6 Exploratpry Data Anal Stats Based EDA 11](#_Toc49724614)

[6.1 Graph based EDA 11](#_Toc49724615)

[7 Stats-Based EDA 15](#_Toc49724616)

[7.1 Technical solution design 15](#_Toc49724617)

[8 Feature Engineering 16](#_Toc49724618)

[8.1 Splitting the data into predictors and target 16](#_Toc49724619)

[8.2 Handling imbalance data(Oversampling) 16](#_Toc49724620)

[8.3 Scaling the data(predictors) 17](#_Toc49724621)

[9 ML Model Selection 18](#_Toc49724622)

[9.1 Technical solution design 18](#_Toc49724623)

[10 Model Tuning and Optimization 19](#_Toc49724624)

[10.1 Technical solution design 19](#_Toc49724625)

[11 Prediction Pipeline 21](#_Toc49724626)

[11.1 Technical solution design 21](#_Toc49724627)

[12 UI Development 23](#_Toc49724628)

[13 Dockerization: 26](#_Toc49724629)

[14 Deployment to cloud: 27](#_Toc49724630)

[15 Challenges: 31](#_Toc49724631)

[16 Learnings: 32](#_Toc49724632)

# Introduction

## Problem Statement:

The goal here is to build an end to end automated Machine Learning solution to evaluate the condition of the car with the given features and factors of the car.

To build and deploy a web application where the car details are entered into a web-based form which then outputs a predicted car evaluation decision in terms of anaccounted,accounted, goo or very good.

A good used car can be evaluated based on the following factors

Price, maintance cost, No.of doors,Capacity,size of luggage boot

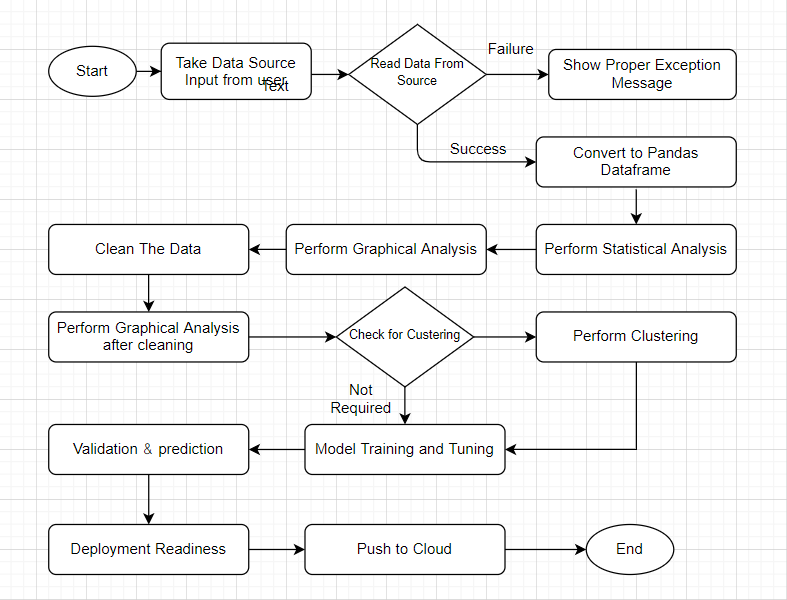
And safety

The machine learning solution was created for the analysis fo the car evaluation data set from UCI machine learning Library.

## Objectives:

1. Enable reading/loading of data from the various sources and convert them into pandas dataframe
2. Perform statistical analytics of the data and prepare a table for the analysis and show it on screen.
3. Perform graphical analysis for the data and Showcase the results (graphs) on the screen.
4. Perform data cleaning operation with all the steps required and showcase a report on screen.
5. After data cleaning showcase the graphical analysis once again for comparison.
6. Check whether clustering is required or not.
7. Choose the appropriate ML model for training.
8. Perform model Tuning.
9. Create a list of top 3 models and show multiple metrics for them.
10. Give option for prediction.
11. Give options for docker container creation.
12. Give option for automatic cloud deployment.

# Workflow Overall:



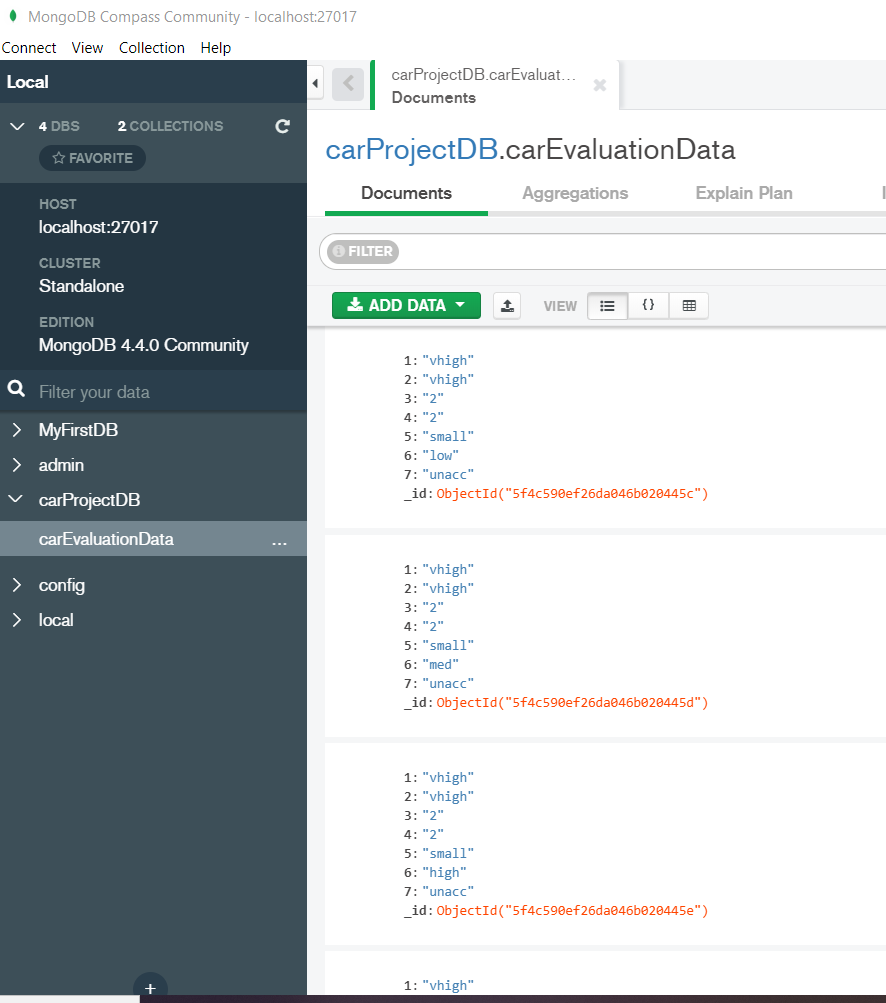
# Libraries used

* NUMPY
* PANDAS
* SKLEARN
* SEABORN
* MATPLOTLIB
* FLASK
* REQUESTS
* Pymongo
* Smote
* Flaskdashboard

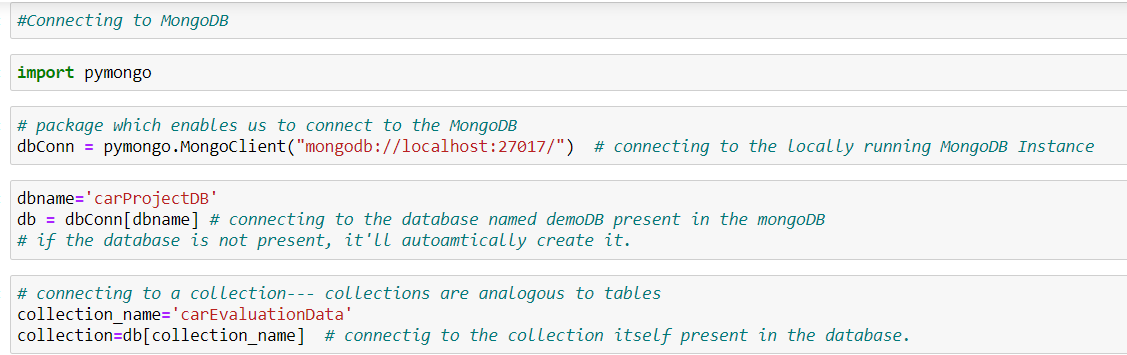
# Data Preprocessing

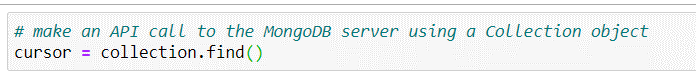
## Import dataset from Database(MongoDB):

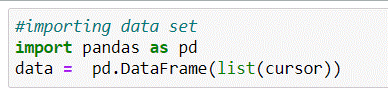
Initially the data was downloaded from UCI website stored in MongoDB database as shown below:

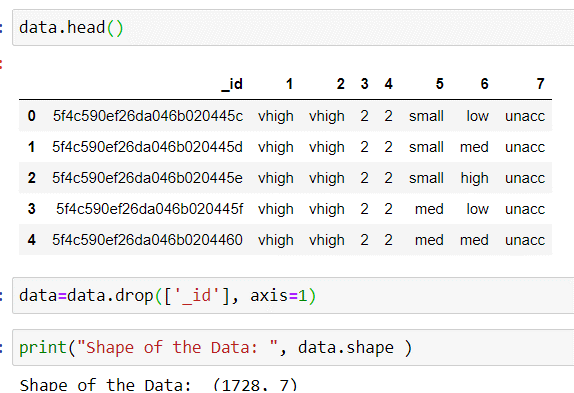


Reading the data as csv file from MongoDB to python:

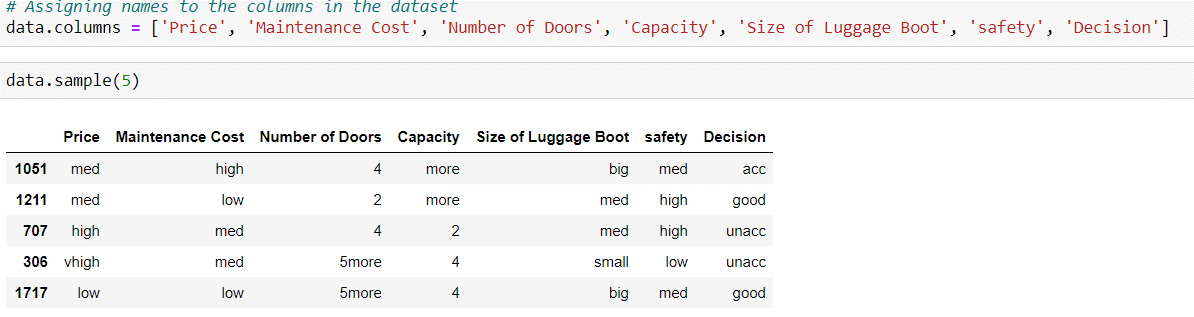




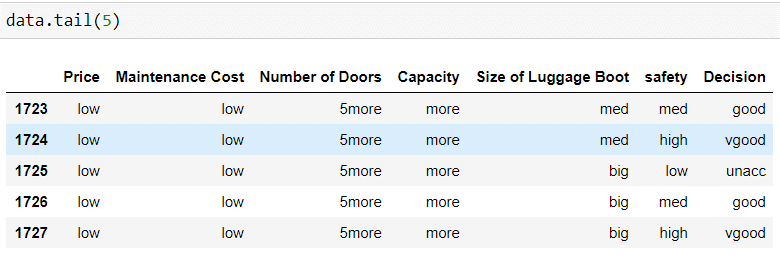


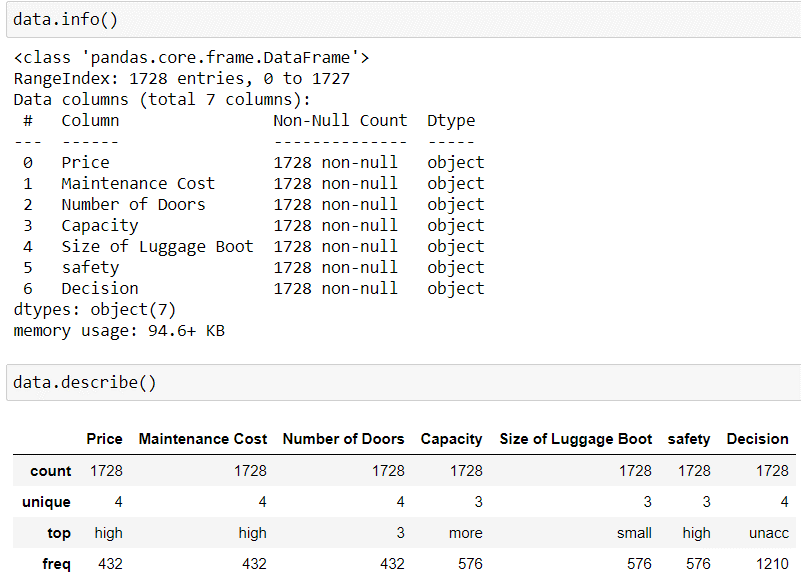


Renaming the columns:



## data insights:





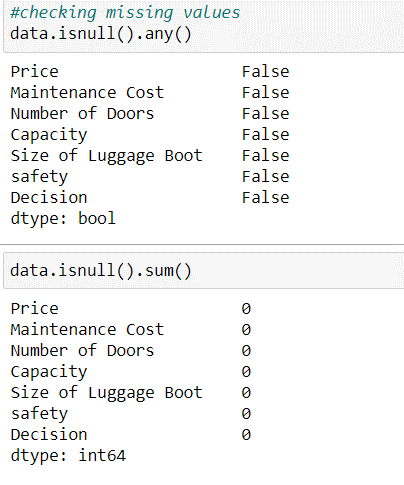
Observation:

1.There are 1728 records and 7 columns in the dataset

2.many of the columns are categorical

3.Decision is a Target variable and it is categorical with four unique values for it.So the machine learning model will be a multi-variate classification problem

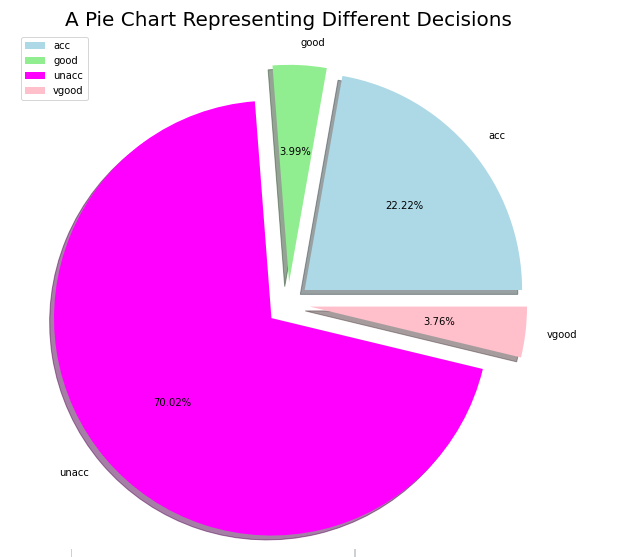
## Checking missing values:



There are no missing values in the data set.

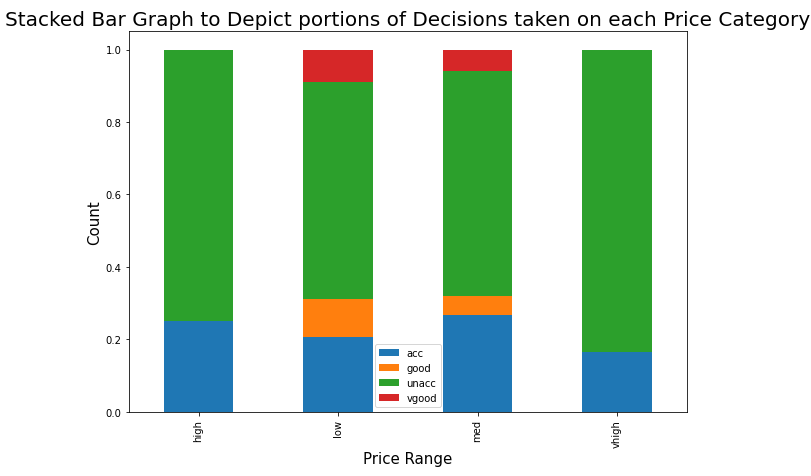
# Exploratpry Data Anal Stats Based EDA

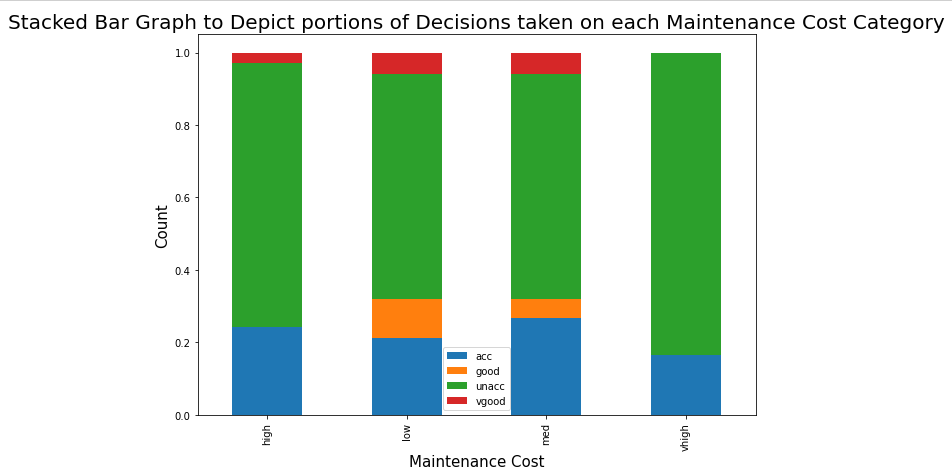
## Graph based EDA

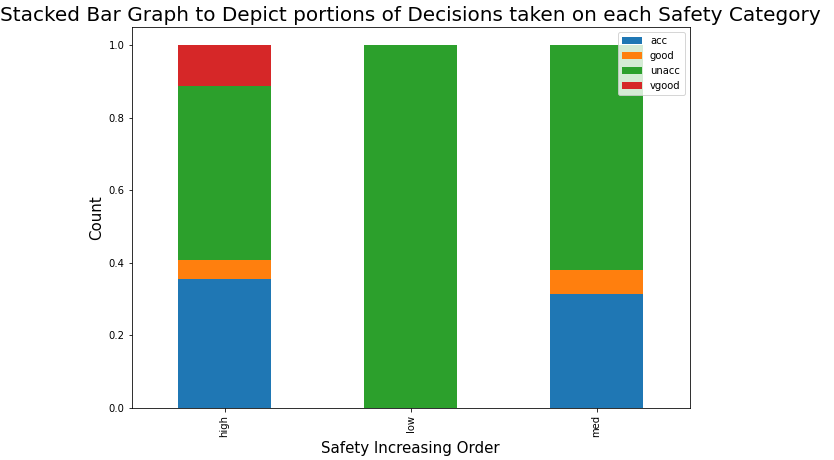


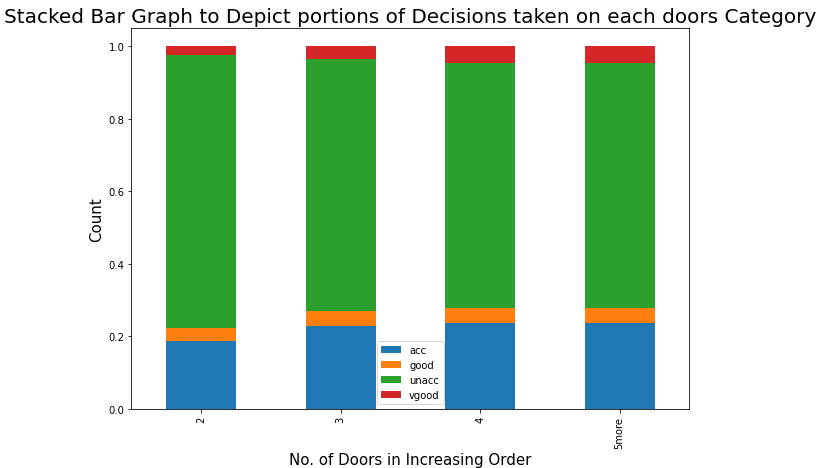
Most cars in the garage are unaccounted and very less number of them are in very good condition.

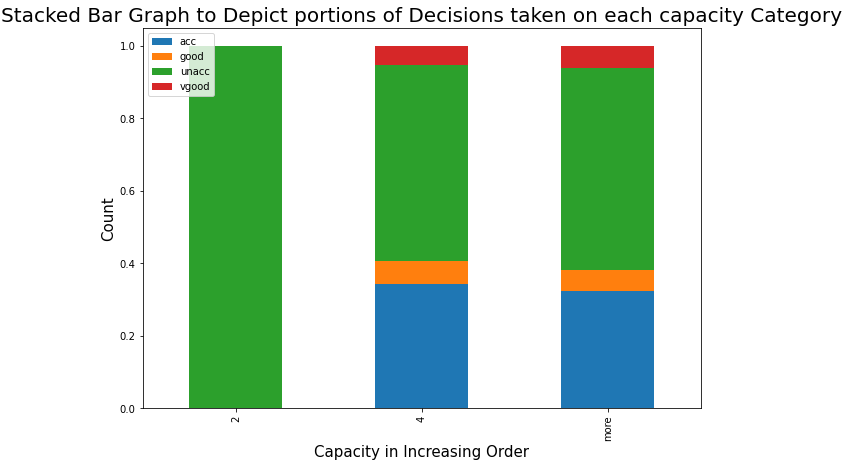
This shows the data here looks heavily imbalanced.

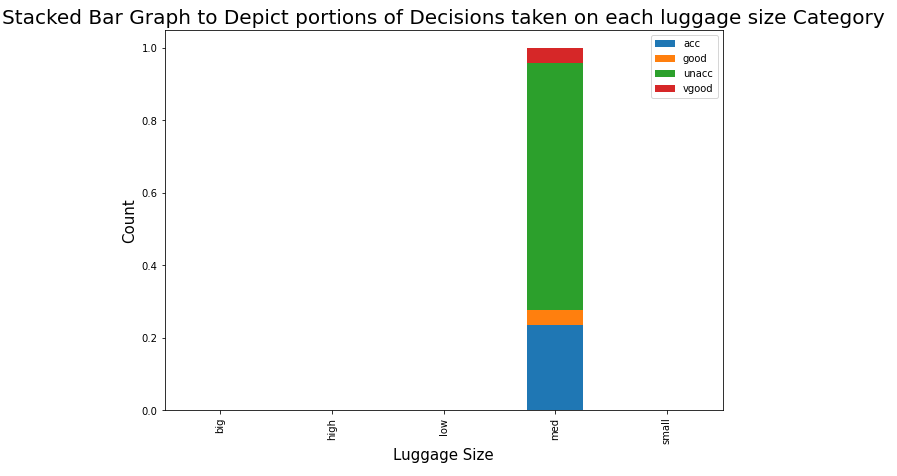








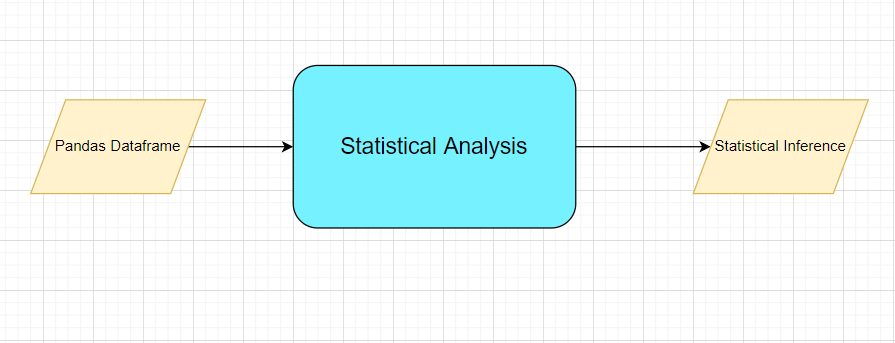


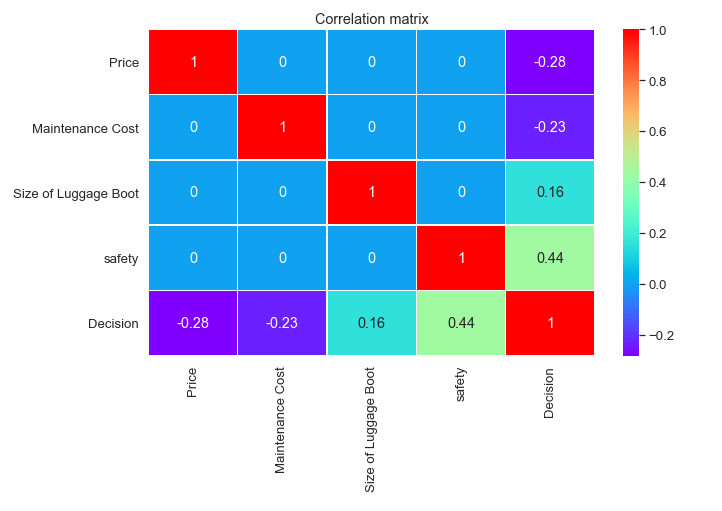


# Stats-Based EDA

Co-relation matix with heat map was plotted to show the co-relation between features.

## Technical solution design





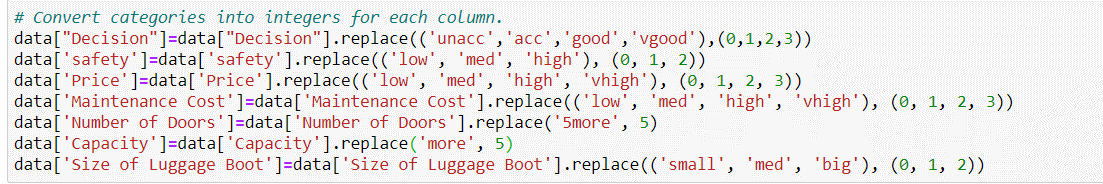
“Safety “ has the highest co-relation with the target variable”Decision”,this shows that Safety is an important predictor for the Target variables.

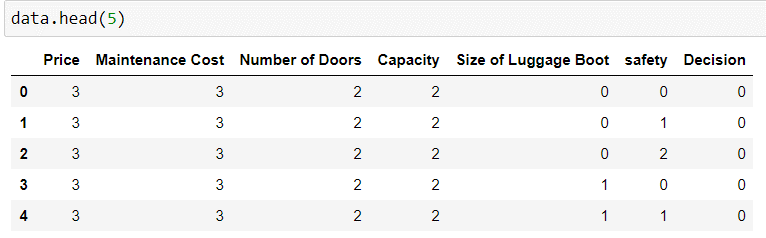
The co-relation between other predictors are very small ,this indicates there is no “multi-co-linearity”

# Feature Engineering

Data set contains categorical values in the columns.In order build the machine learning model, we need to convert these categorical values into numerical values.

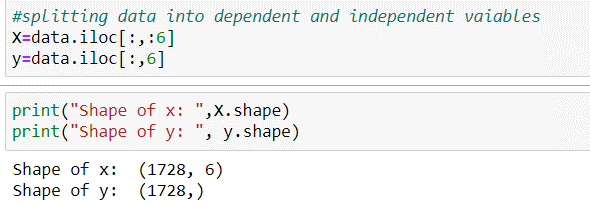
In this the categorical columns are ordinal, so Label encoding has been used to convert the categorical values to numerical values.





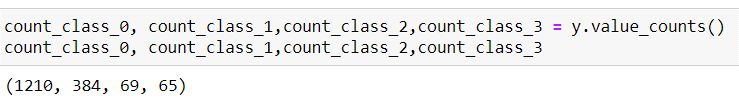
Now the values in the data set are numerical.

## Splitting the data into predictors and target



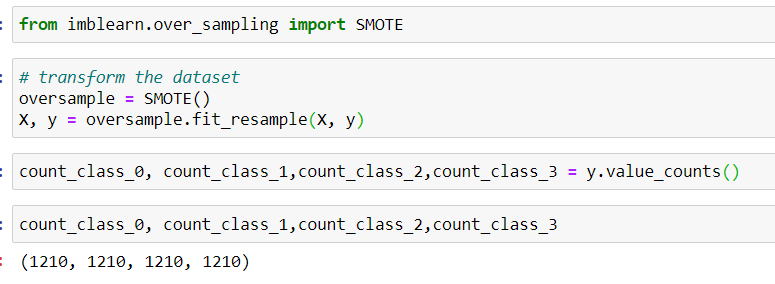
## Handling imbalance data(Oversampling)

Here the original dataset is highly imbalanced



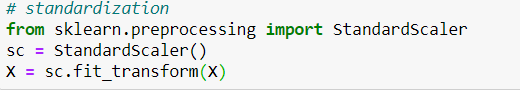
So need to balance the data in order to get the reliable accuracy of the ML model.

Used the SMOTE over sampling method , to balance the minority and majority classes.



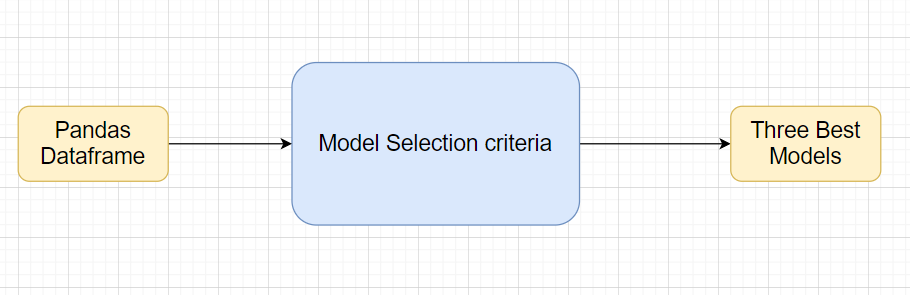
## Scaling the data(predictors)

Now ,the data set has to be scaled by using standard scaling method to make the values are in the same range.

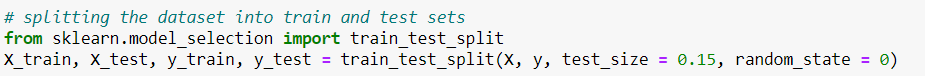


# ML Model Selection

## Technical solution design



Split the entire data set into train and validation sets.



In this section we build and try 6 models:

Logistic Regression

K-nearest Neighbour

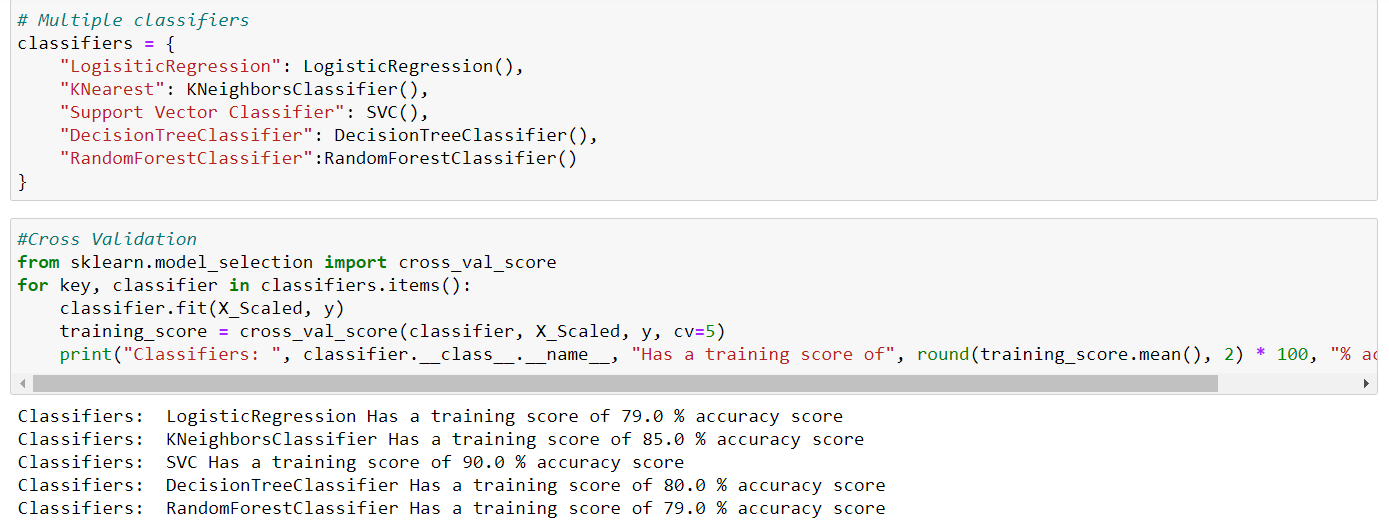
Support Vector classification

Naive bayees

Decision tree

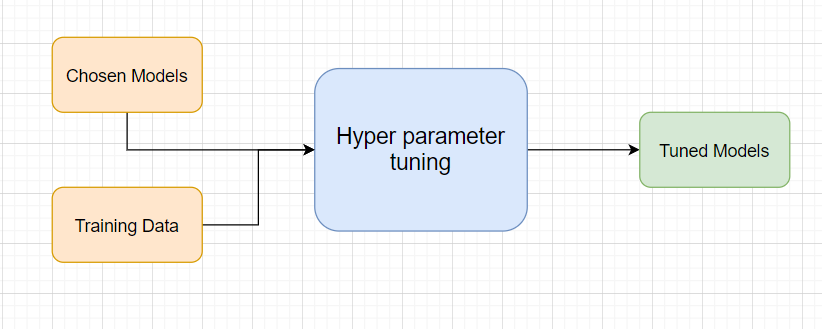
Random Forest

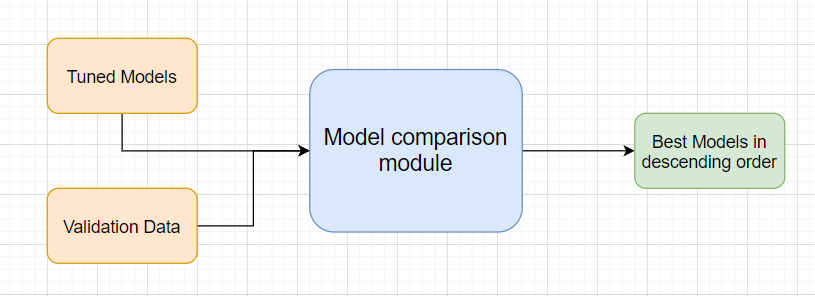
Each model will be trained and make a prediction for the test set. Accuracy, f1 score, confusion matrix and ROC will be calculated for each model. Then we will use the GridSearchCV to tune the models and search for the best hyperparameters in order to increase the accuracy of each model.

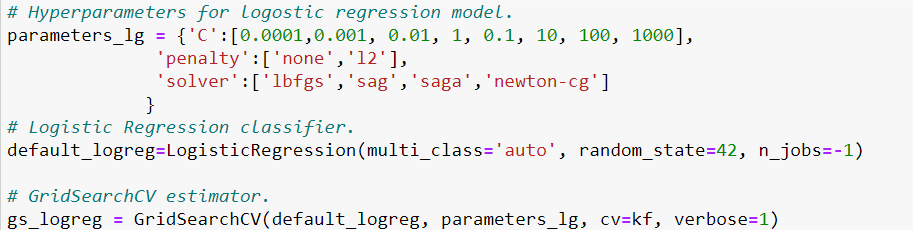


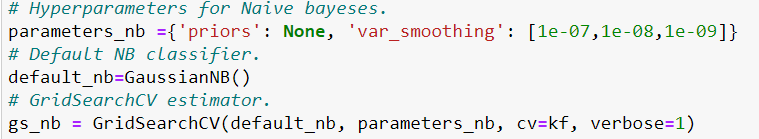
# Model Tuning and Optimization

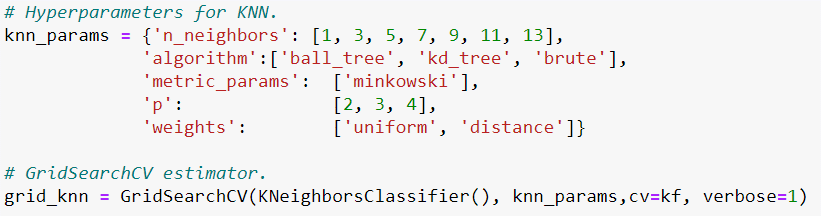
## Technical solution design

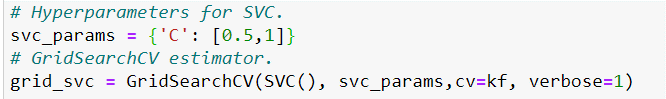


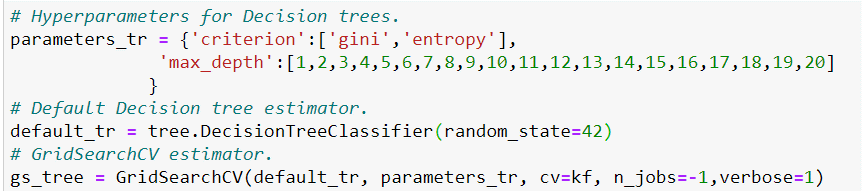


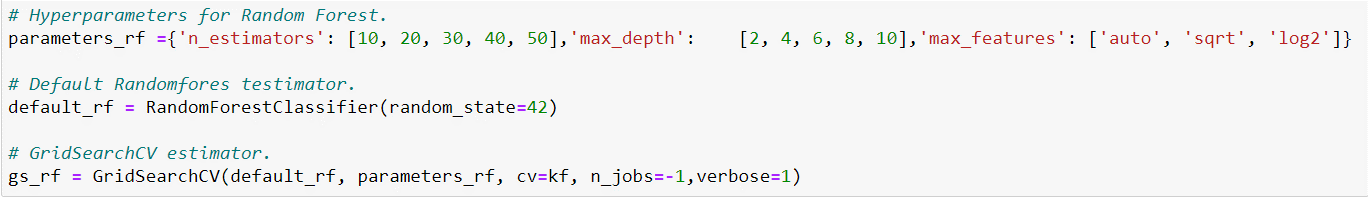


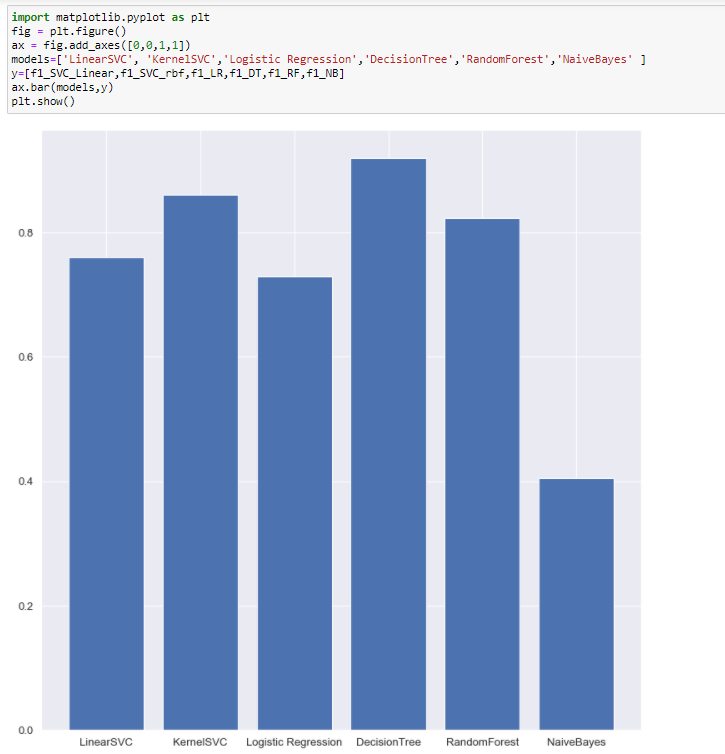










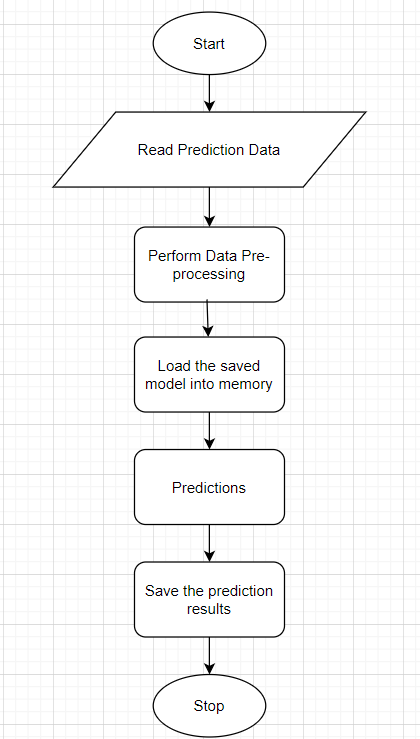


Decision tree model is over predicting .

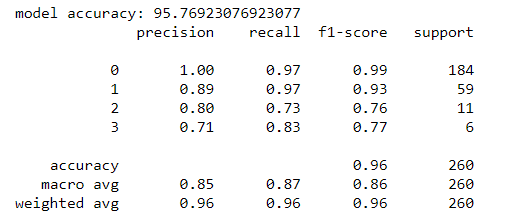
Since the SVC model gives the higher accuracy , the final model chosen is the support vector machine.

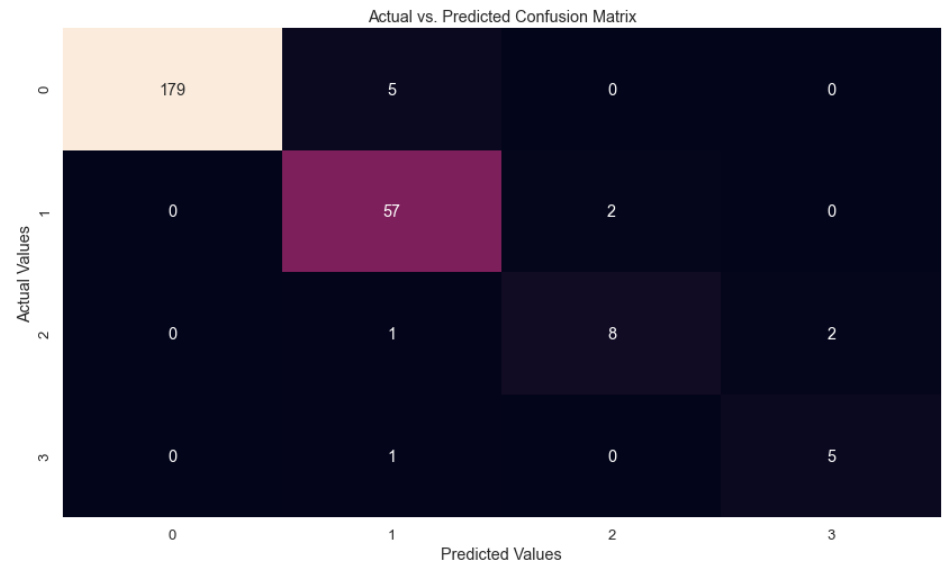
# Prediction Pipeline

## Technical solution design

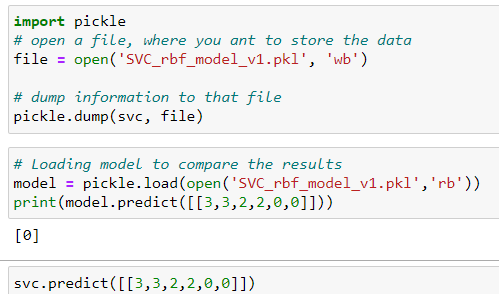


**Final prediction by using SVC model:**





Picle the python code to predict the test data.

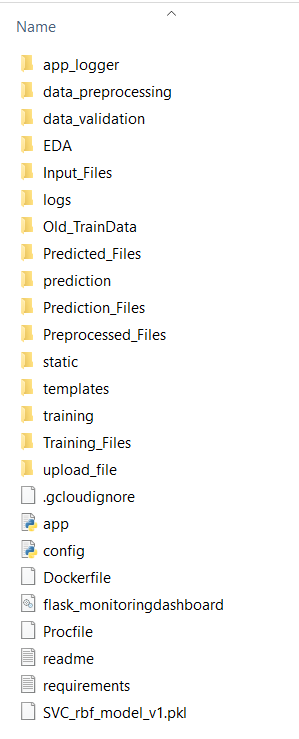


# UI Development

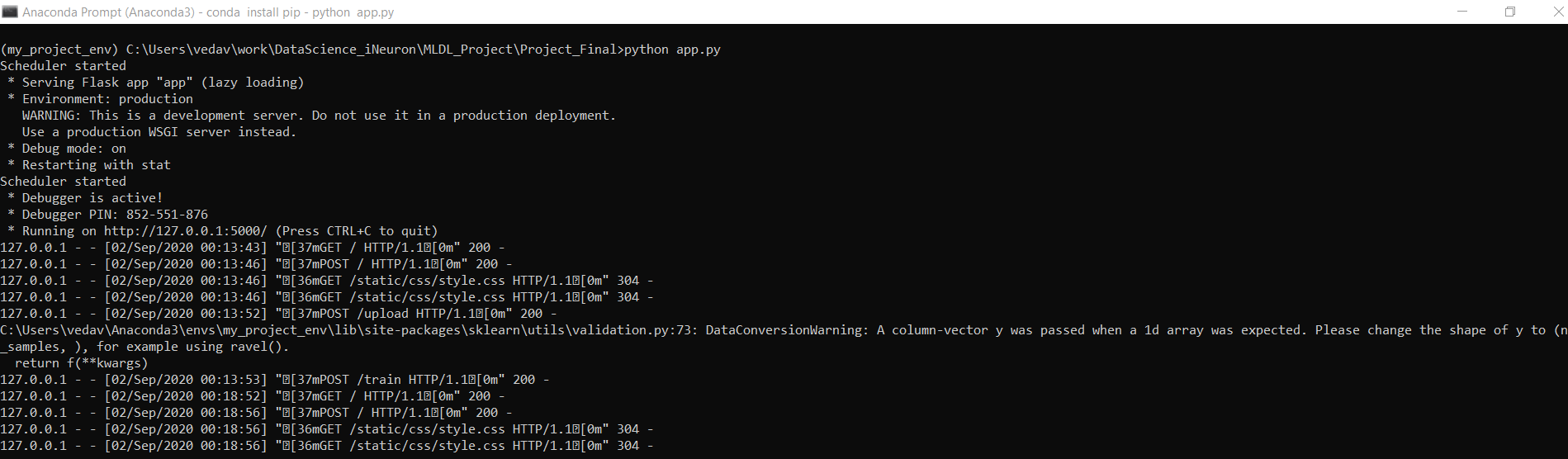
We have created the web application using Flask framework in Python.

Run the FLASK app and get the URL of the web app on the local machine.

These are all the final files required for UI develpment



Run the FLASK app in Anaconda Prompt to test it locally.

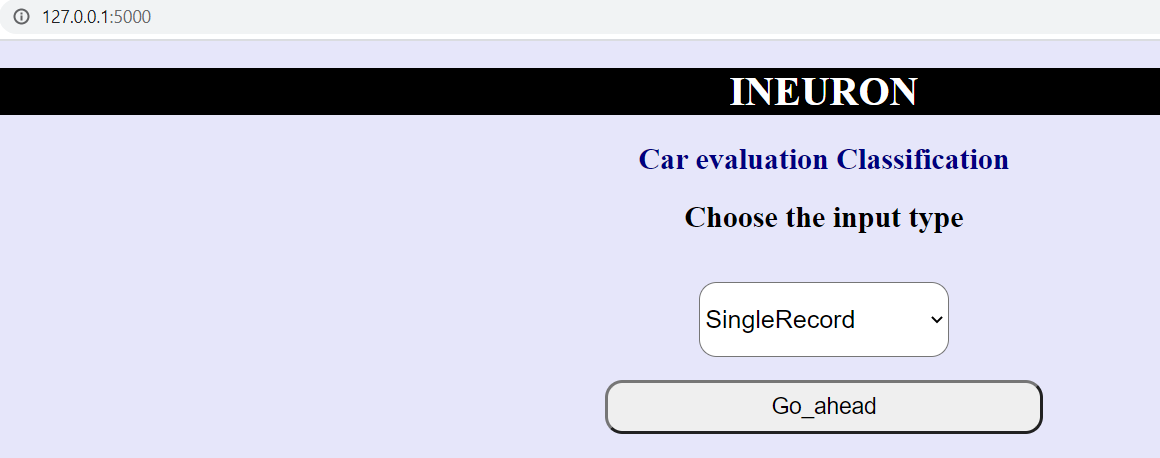


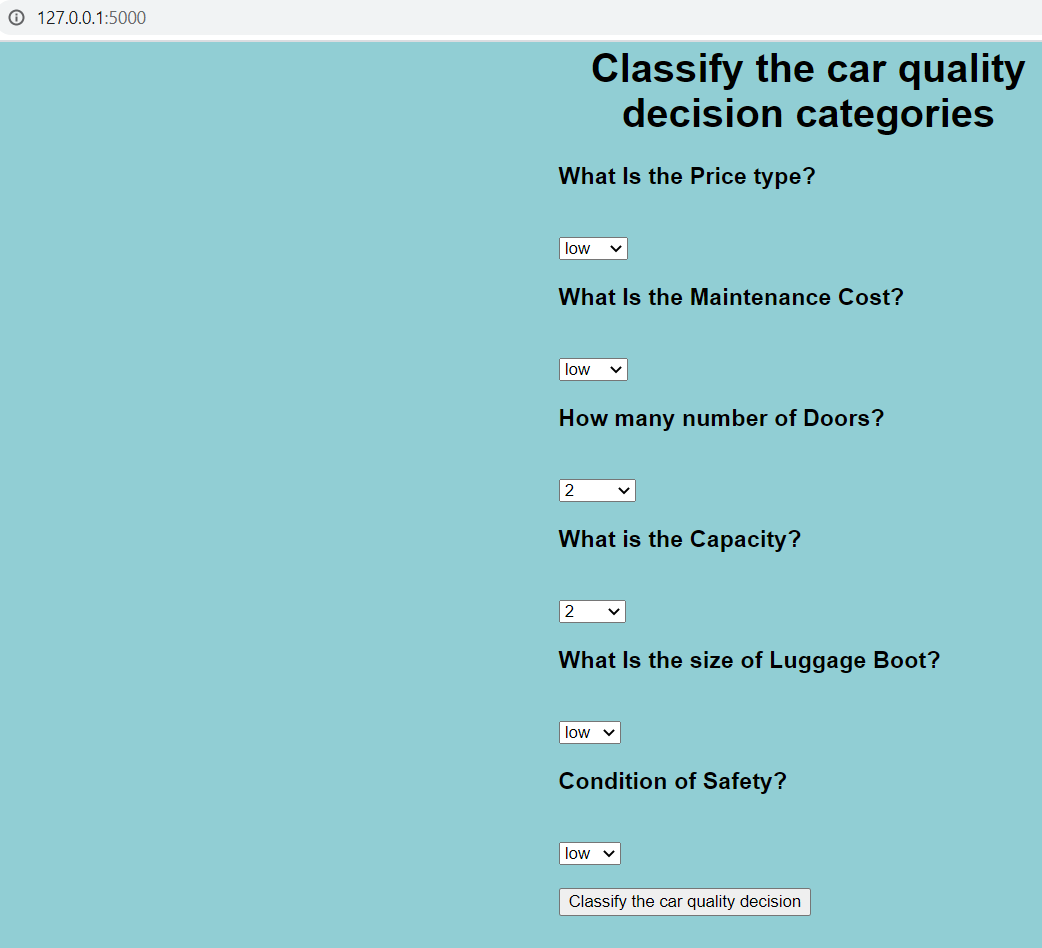
The URL is http://127.0.0.1:5000/

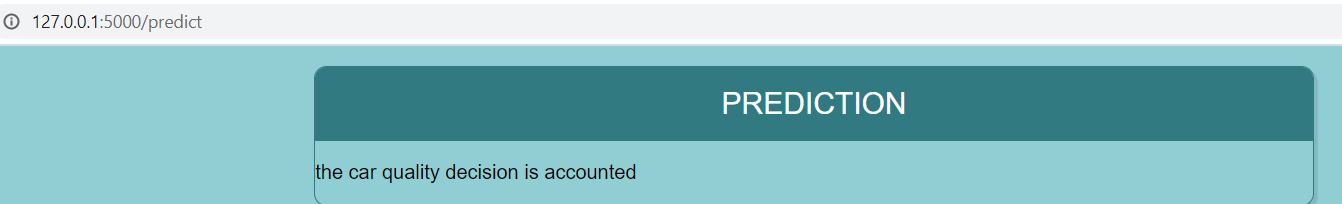
Open this URL in a browser to see the web app , This is how our web application looks:

It has both the options of single value input and bulk upload as csv .

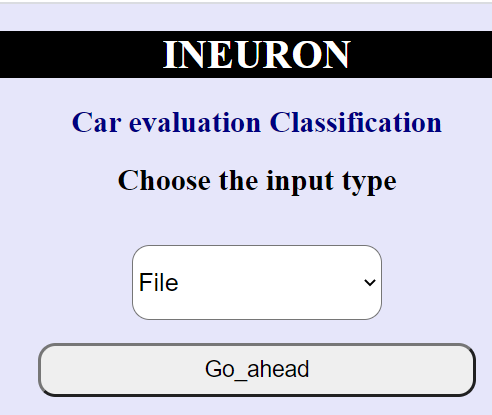
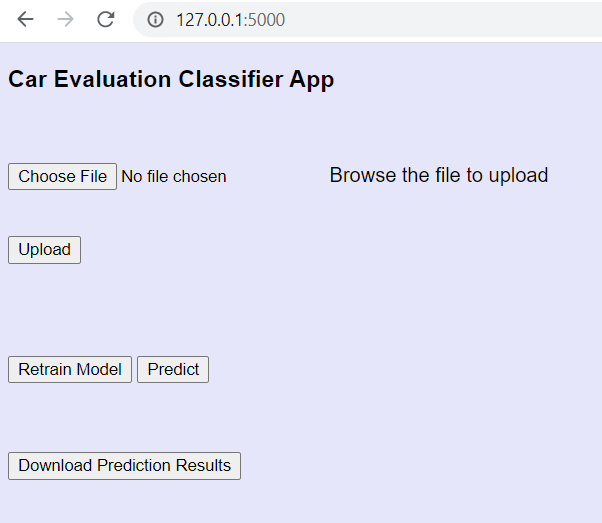
For Single record entry,UI shown below:

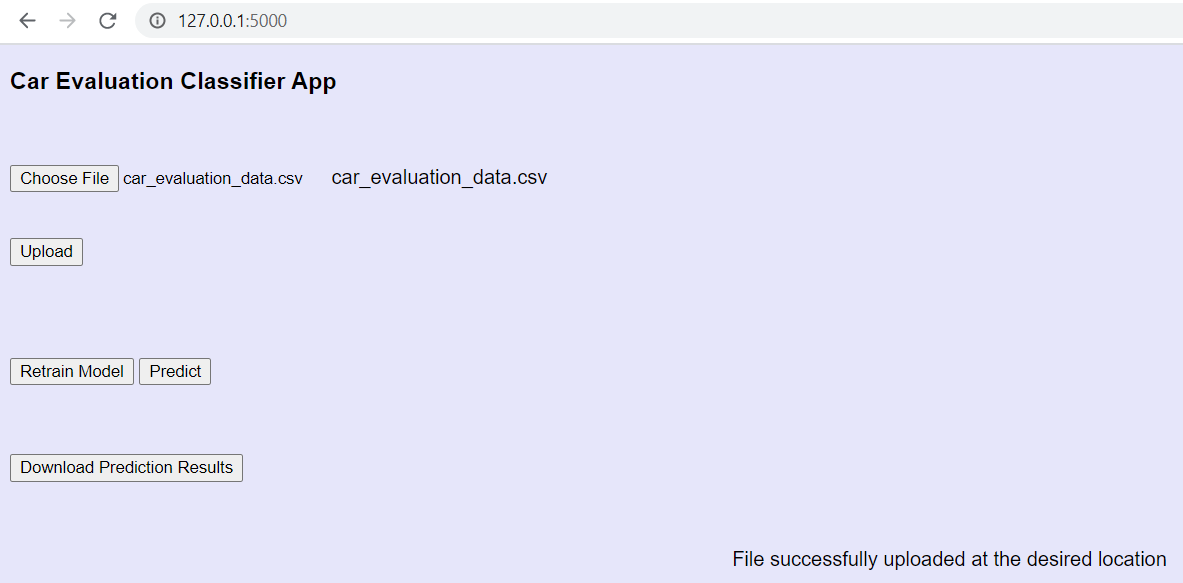




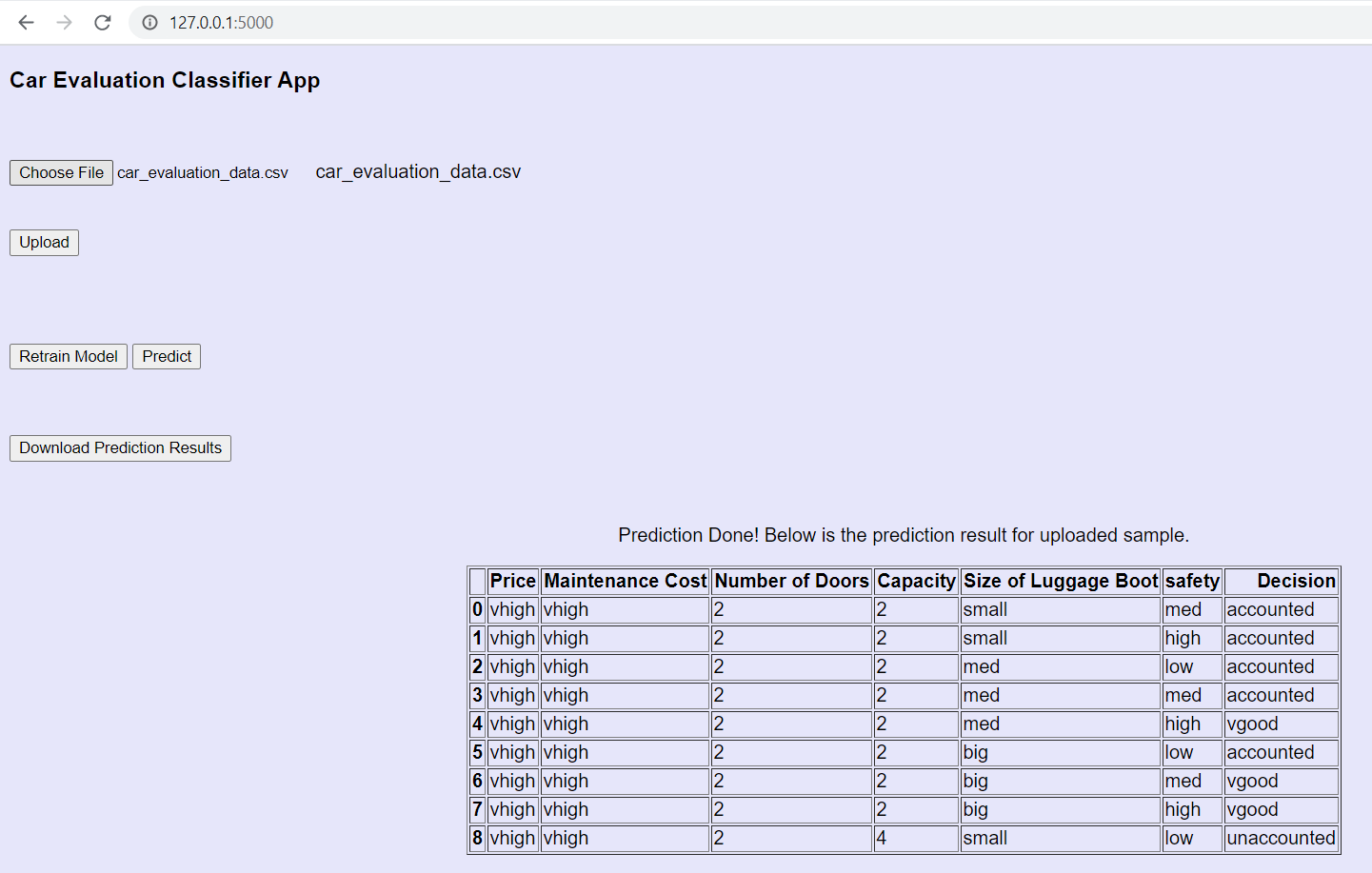


For bulk upload entry,UI shown below:



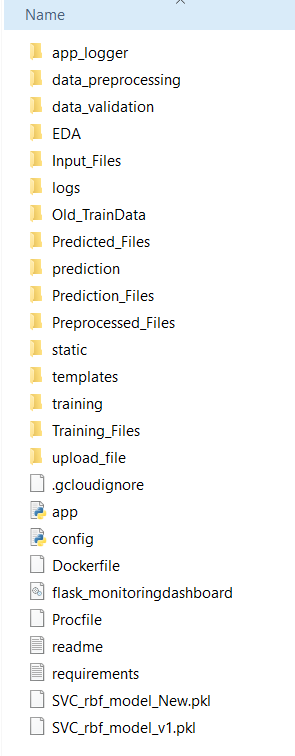
For Prediction



UI For Retraining:





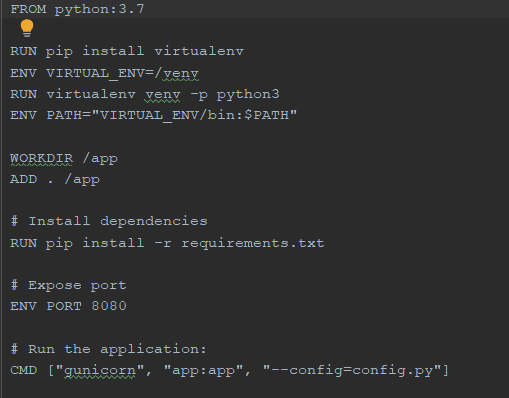
 After Retraining, New pickle file has been saved

# Dockerization:

**container**is a type of software that packages up an application and all its dependencies so the application runs reliably from one computing environment to another.

Here Docker is used to build ,run and manage containers.

Create a docker file and build to create a container



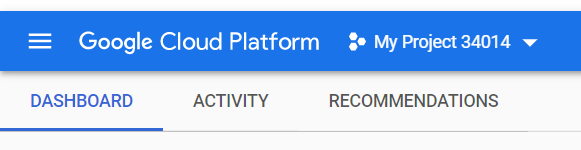
Along with Docker , I used the Kubernetes to create the cluster.

Kubernetes is a system for running and coordinating containerized applications across a cluster of machines. It is a platform designed to completely manage the life cycle of containerized applications.

# Deployment to cloud:

Deployed the app in GCP cloud as shown below:

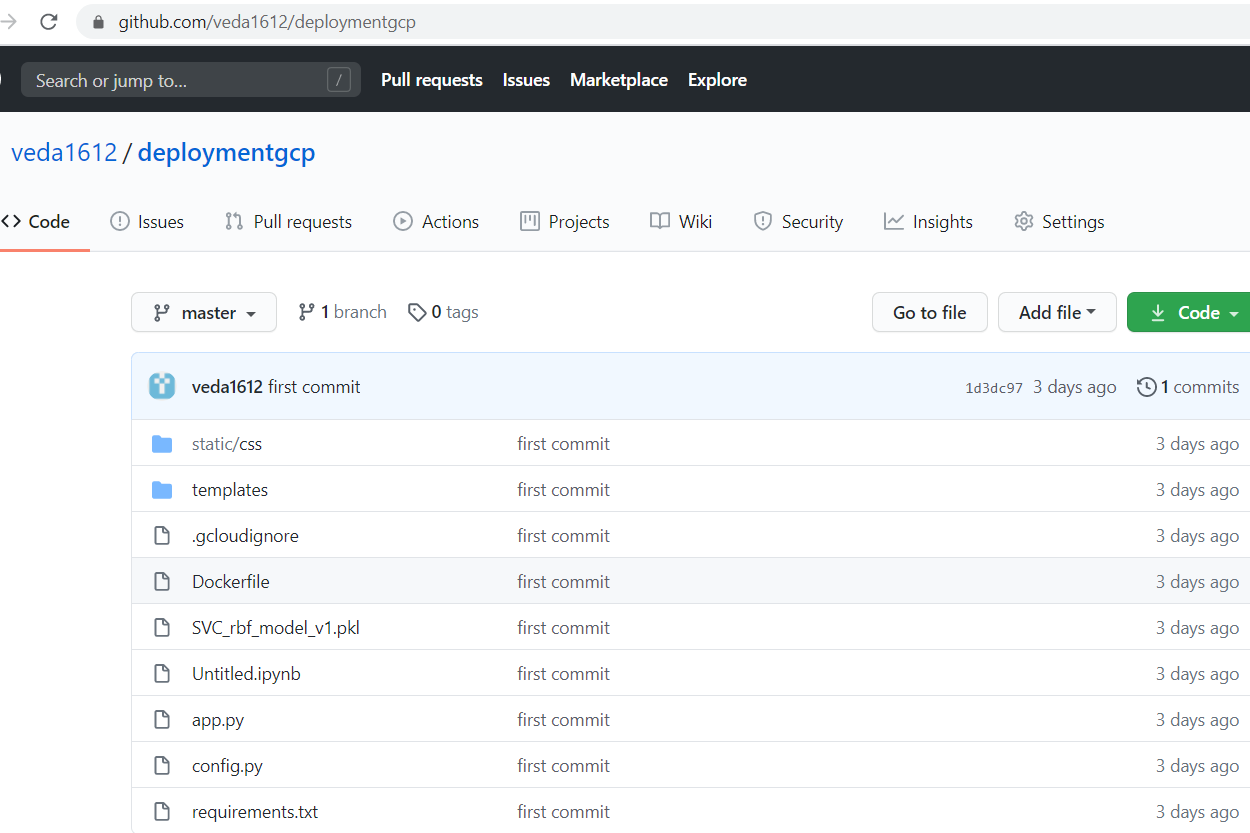
**Step1:Create a new project in GCP Console**



**Step2:Import Project Code:**

Before pushing it to GCP, I pushed the required files in Github at this location:

***git clone https://github.com/veda1612/deploymentgcp.git***



Login to GCP and push the above files from GithuB to GCP.

Created a project with some default name:



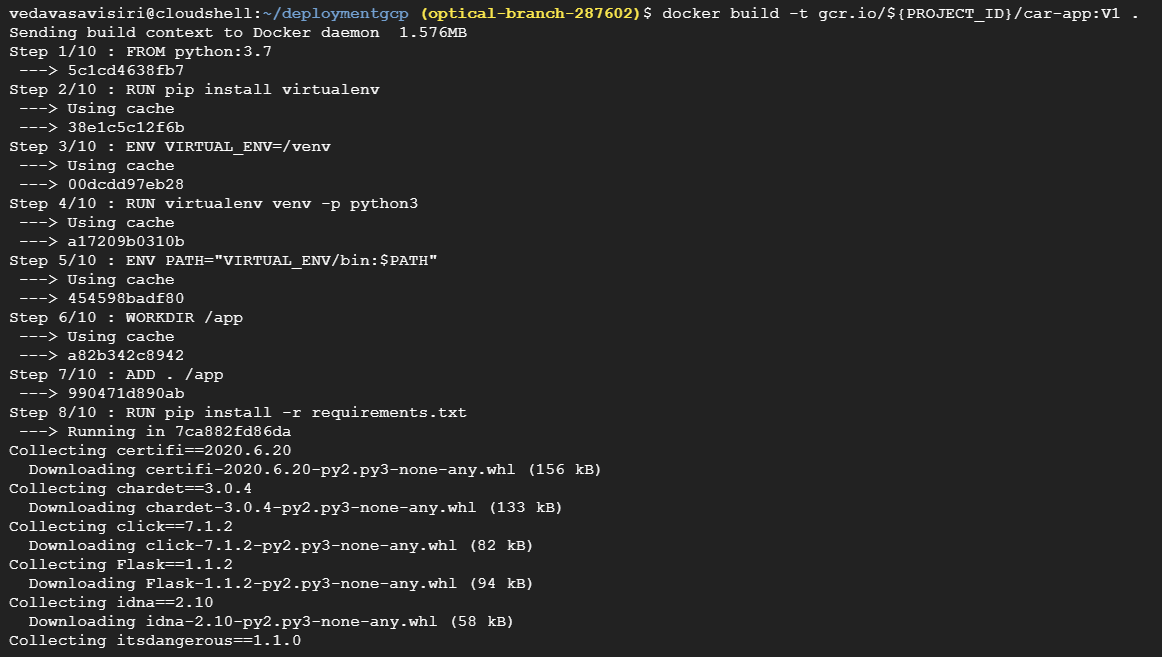
**Step 3— Set Project ID Environment Variable**

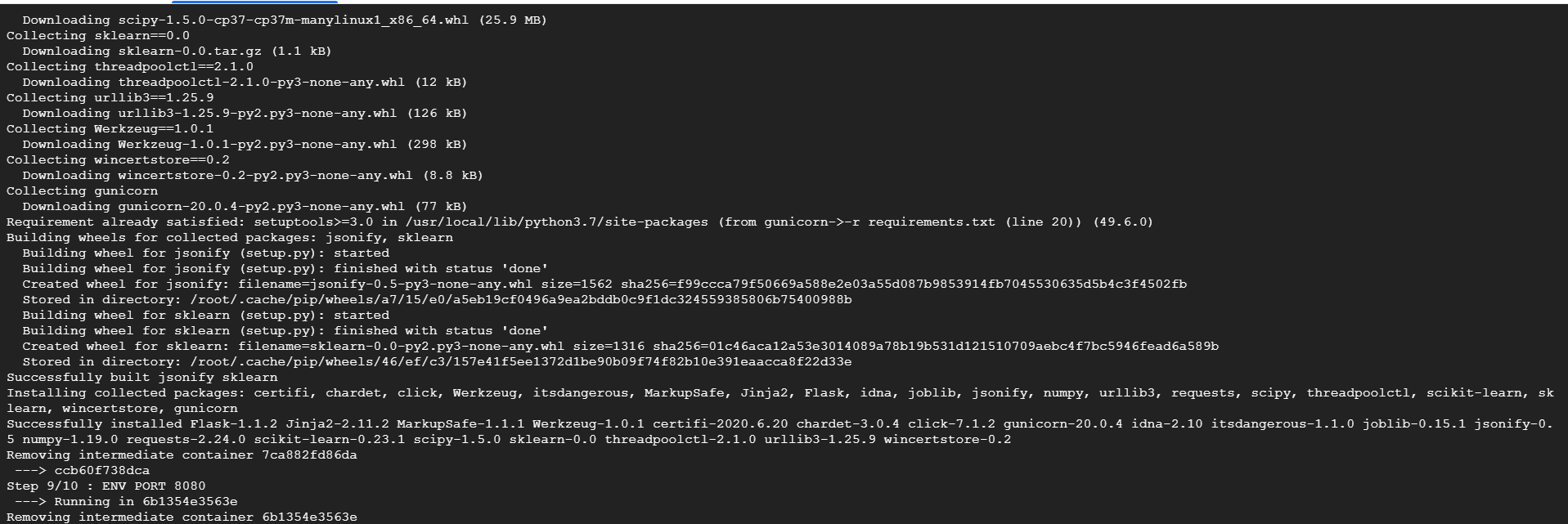
***export PROJECT\_ID=optical-branch-287602***



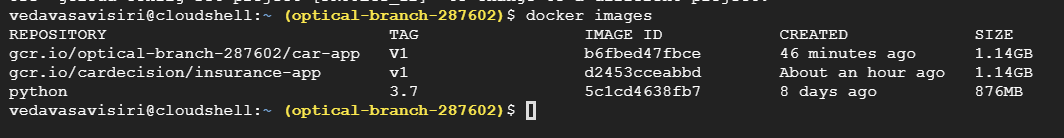
**Step 4— Build the docker image**

Build the docker image:

***docker build -t gcr.io/${PROJECT\_ID}/car-app:V1 .***



Check the docker images:

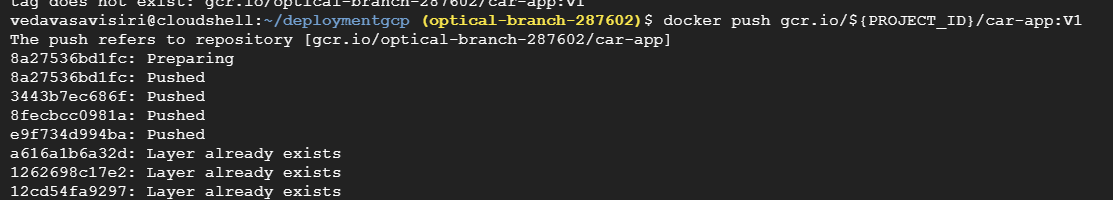


**Step 5— Upload the container image**

***gcloud auth configure-docker***

***docker push gcr.io/${PROJECT\_ID}/cap-app:V1***

upload the docker image to Google Container Registry:



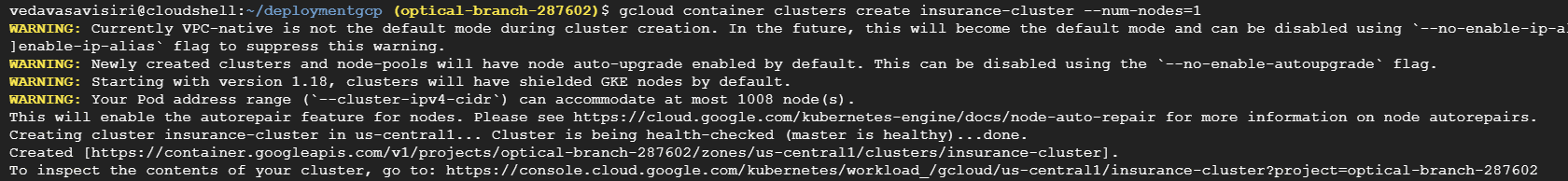
**Step 6— Create Cluster:**

**Set project ID and Compute Engine zone**

***gcloud config set project $PROJECT\_ID   
gcloud config set compute/zone us-central1***

***Create a cluster by executing the following code:***

***gcloud container clusters create car-cluster --num-nodes=2***



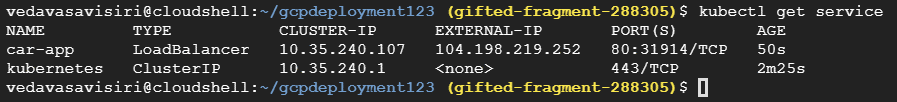
**Step 7— Deploy Application:**

***kubectl create deployment car-app --image=gcr.io/${PROJECT\_ID}/cap-app:V1***

**Step 8— Expose your application to the internet**

***kubectl expose deployment cap-app --type=LoadBalancer --port 80 --target-port 8080***

**Step 9— Check Service**

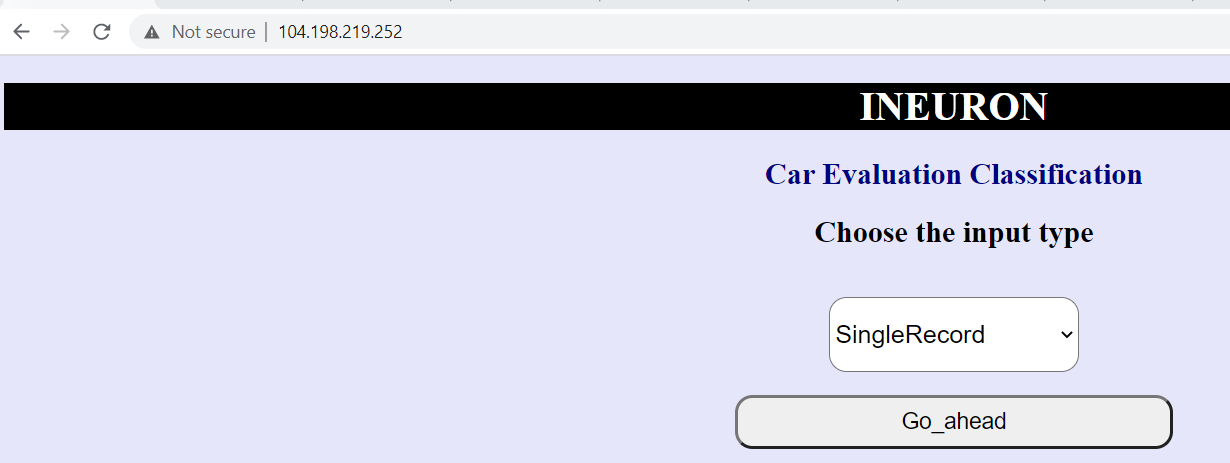


Open the above URL to see the app and test it.

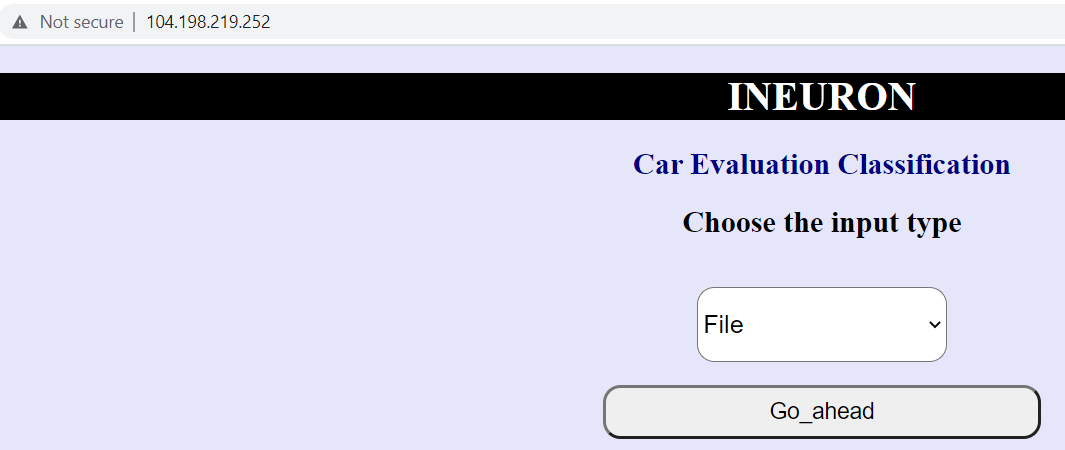
**Step 10— see the app:**

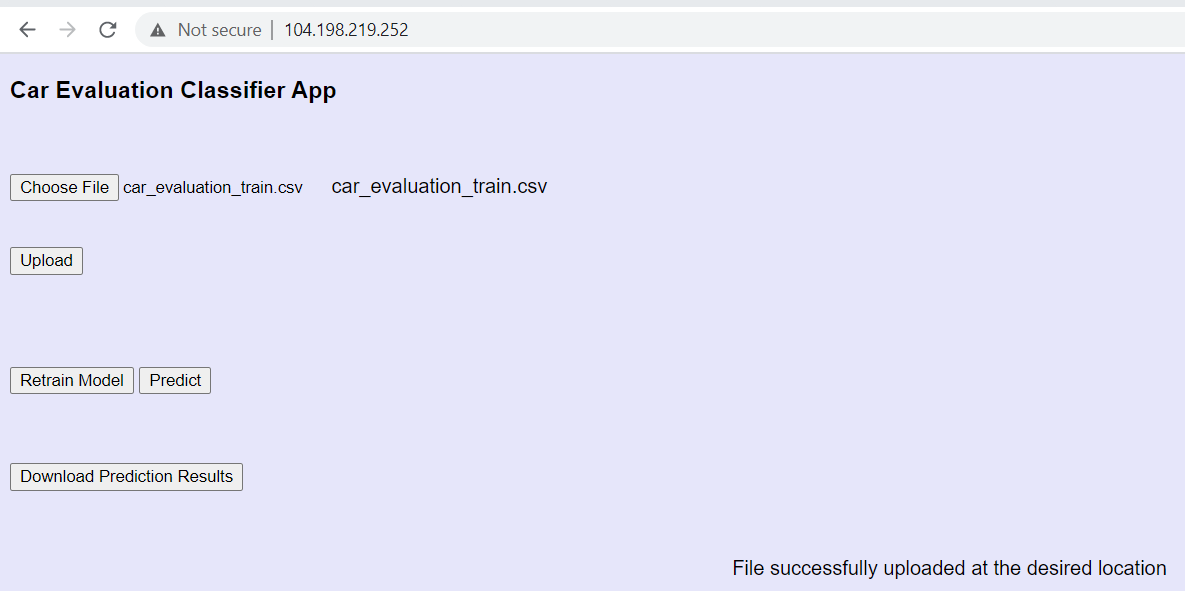
<http://104.198.219.252/>

For single record prediction



For Bulk upload





References:

<https://towardsdatascience.com/deploy-machine-learning-model-on-google-kubernetes-engine-94daac85108b>

# Challenges:

How to implement exception handling, logger files was bit challenging for me

Deployement in GCP was also challenging for me initially,but I had refered the deployment in GCP documentation from iNeuron, then it was very useful to deploy.

# Learnings:

* Machine Learning Fundamentals and Libraries
* Using MongoDB ,pymongo
* Understanding of Flask Framework
* HTML
* CSS
* Docker
* Deployment,Google Cloud Platform

**THANK YOU**

**From**

**Veda**

**Group:1I**

***Github link: https://github.com/veda1612/gcpdeployment123.git***