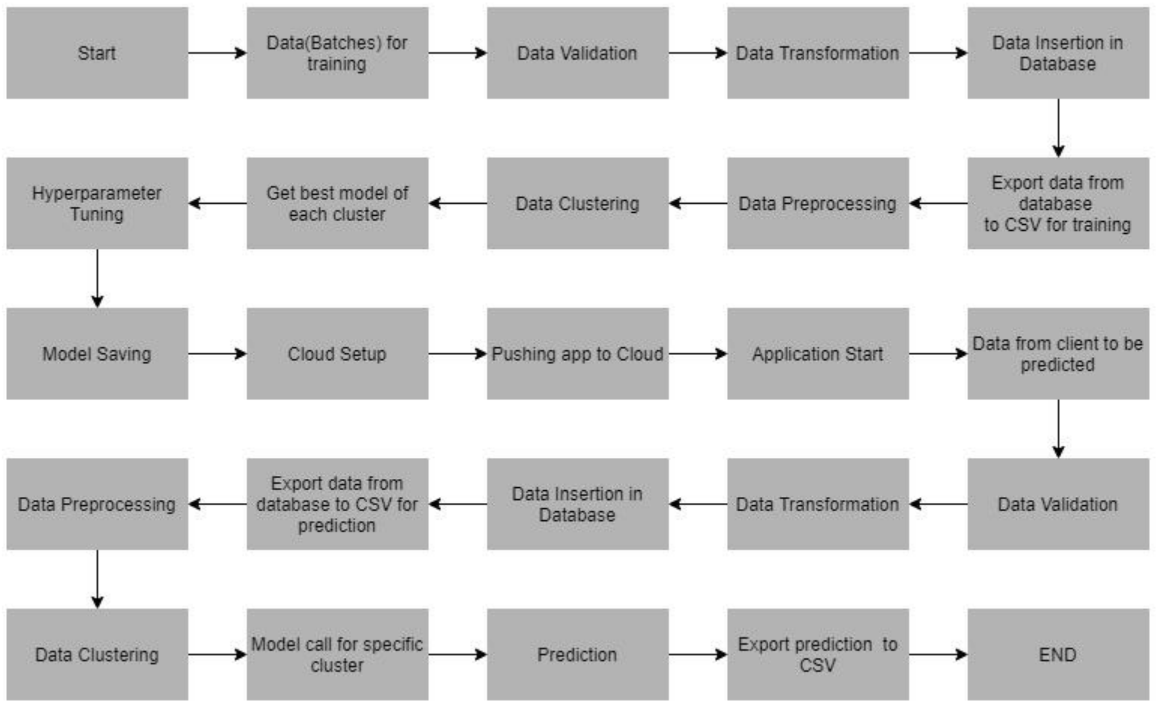
Problem Statement

To build a regression model to predict the compressive strength of concrete based on the different features in the training data.

Architecture


Start
Data(Batches) for training
Data Validation
Data Transformation
Data Insertion in Database
Export data from database to CSV for training
Data Preprocessing
Data Clustering
Get best model of each cluster
Hyperparameter Tuning
Model Saving
Cloud Setup
Pushing app to Cloud
Application Start
Data from client to be predicted
Data Validation
Data Transformation
Data Insertion in Database
Export data from database to CSV for prediction
Data Preprocessing
Data Clustering
Model call for specific cluster
Prediction
Export prediction to CSV
END

Data Description

Given is the variable name, variable type, the measurement unit and a brief description.

The concrete compressive strength is the regression problem. The order of this listing corresponds to the order of numerals along the rows of the database.

Name	Data Type	Measurement	Description
Cement (component 1)	quantitative	kg in a m3 mixture	Input Variable
Blast Furnace Slag (component 2)	quantitative	kg in a m3 mixture	Input Variable-- Blast furnace slag is a nonmetallic coproduct produced in the process. It

			consists primarily of silicates, aluminosilicates, and calcium-alumina-silicates
Fly Ash (component 3)	quantitative	kg in a m3 mixture	Input Variable- it is a coal combustion product that is composed of the particulates (fine particles of burned fuel) that are driven out of coal-fired boilers together with the flue gases.
Water (component 4)	quantitative	kg in a m3 mixture	Input Variable
Superplasticizer (component 5)	quantitative	kg in a m3 mixture	Input Variable-- Superplasticizers (SP's), also known as high range water reducers, are additives used in making high strength concrete. Their addition to concrete or mortar allows the reduction of the water to cement ratio without negatively affecting the workability of the mixture, and enables the production of self-consolidating concrete and high performance concrete
Coarse Aggregate (component 6)	quantitative	kg in a m3 mixture	Input Variable-- construction aggregate, or simply "aggregate", is a broad category of coarse to medium grained particulate material used in construction, including sand, gravel, crushed stone, slag, recycled concrete and geosynthetic aggregates
Fine Aggregate (component 7)	quantitative	kg in a m3 mixture	Input Variable—Similar to coarse aggregate, the constitution is much finer.

Age	quantitative	Day (1~365)	Input Variable
Concrete compressive strength	quantitative	MPa	Output Variable

Apart from training files, we also require a "schema" file from the client, which contains all the relevant information about the training files such as:

Name of the files, Length of Date value in FileName, Length of Time value in FileName, Number of Columns, Name of the Columns, and their datatype.

Data Validation

In this step, we perform different sets of validation on the given set of training files.

1. Name Validation- We validate the name of the files based on the given name in the schema file. We have created a regex pattern as per the name given in the schema file to use for validation. After validating the pattern in the name, we check for the length of date in the file name as well as the length of time in the file name. If all the values are as per requirement, we move such files to “Good_Raw” folder else we move such files to “Bad_Raw” folder.

2. Number of Columns - We validate the number of columns present in the files, and if it doesn't match with the value given in the schema file, then the file is moved to “Bad_Raw” folder.

3. Name of Columns - The name of the columns is validated and should be the same as given in the schema file. If not, then the file is moved to “Bad_Raw” folder.

4. The datatype of columns - The datatype of columns is given in the schema file. This is validated when we insert the files into Database. If the datatype is wrong, then the file is moved to “Bad_Raw” folder.

5. Null values in columns - If any of the columns in a file have all the values as NULL or missing, we discard such a file and move it to “Bad_Raw” folder.

Data Insertion in Database

1) Database Creation and connection - Create a database with the given name passed. If the database is already created, open the connection to the database.

2) Table creation in the database - Table with name - "Good_Raw_Data", is created in the database for inserting the files in the “Good_Raw” folder based on given column names and datatype in the schema file. If the table is already present, then the new table is not created and new files are inserted in the already present table as we want training to be done on new as well as old training files.

3) Insertion of files in the table - All the files in the “Good_Raw” folder are inserted in the above-created table. If any file has invalid data type in any of the columns, the file is not loaded in the table and is moved to “Bad_Raw” folder.

Model Training

1) Data Export from Db - The data in a stored database is exported as a CSV file to be used for model training.

2) Data Preprocessing

a) Check for null values in the columns. If present, impute the null values using the KNN imputer

b) transform the features using log transformation

c) Scale the training and test data separately

3) Clustering - KMeans algorithm is used to create clusters in the preprocessed data. The optimum number of clusters is selected by plotting the elbow plot, and for the dynamic selection of the number of clusters, we are using "KneeLocator" class. The idea behind clustering is to implement different algorithms to train data in different clusters. The Kmeans model is trained over preprocessed data and the model is saved for further use in prediction.

4) Model Selection - After clusters are created, we find the best model for each cluster. We are using two algorithms, "Random forest Regressor" and “Linear Regression”. For each cluster, both the algorithms are passed with the best parameters derived from GridSearch. We calculate the Rsquared scores for both models and select the model with the best score. Similarly, the model is selected for each cluster. All the models for every cluster are saved for use in prediction.

Prediction Data Description

Client will send the data in multiple set of files in batches at a given location. Data will contain 8 columns.

Apart from prediction files, we also require a "schema" file from client which contains all the relevant information about the training files such as:

Name of the files, Length of Date value in FileName, Length of Time value in FileName, Number of Columns, Name of the Columns and their datatype.

Data Validation

In this step, we perform different sets of validation on the given set of prediction files.

1) Name Validation- We validate the name of the files on the basis of given Name in the schema file. We have created a regex pattern as per the name given in schema file, to use for validation. After validating the pattern in the name, we check for length of date in the file name as well as length of time in the file name. If all the values are as per requirement, we move such files to “Good_Raw” folder else we move such files to “Bad_Raw” folder.

2) Number of Columns - We validate the number of columns present in the files, if it doesn't match with the value given in the schema file then the file is moved to “Bad_Raw” folder.

3) Name of Columns - The name of the columns is validated and should be same as given in the schema file. If not, then the file is moved to “Bad_Raw” folder.

4) Datatype of columns - The datatype of columns is given in the schema file. This is validated when we insert the files into Database. If dataype is wrong then the file is moved to “Bad_Raw” folder.

5) Null values in columns - If any of the columns in a file has all the values as NULL or missing, we discard such file and move it to “Bad_Raw” folder.

Data Insertion in Database

1) Database Creation and connection - Create database with the given name passed. If the database is already created, open the connection to the database.

2) Table creation in the database - Table with name - "Good_Data", is created in the database for inserting the files in the “Good_Raw” folder on the basis of given column names and datatype in the

schema file. If table is already present then new table is not created, and new files are inserted the already present table as we want training to be done on new as well old training files.

3) Insertion of files in the table - All the files in the “Good_Raw” folder are inserted in the above-created table. If any file has invalid data type in any of the columns, the file is not loaded in the table and is moved to “Bad_Raw” folder.

Prediction

1) Data Export from Db - The data in the stored database is exported as a CSV file to be used for prediction.

2) Data Preprocessing

   a) Check for null values in the columns. If present, impute the null values using the KNN imputer

   b) transform the features using log transformation

   c) Scale the training and test data separately

3) Clustering - KMeans model created during training is loaded, and clusters for the preprocessed prediction data is predicted.

4) Prediction - Based on the cluster number, the respective model is loaded and is used to predict the data for that cluster.

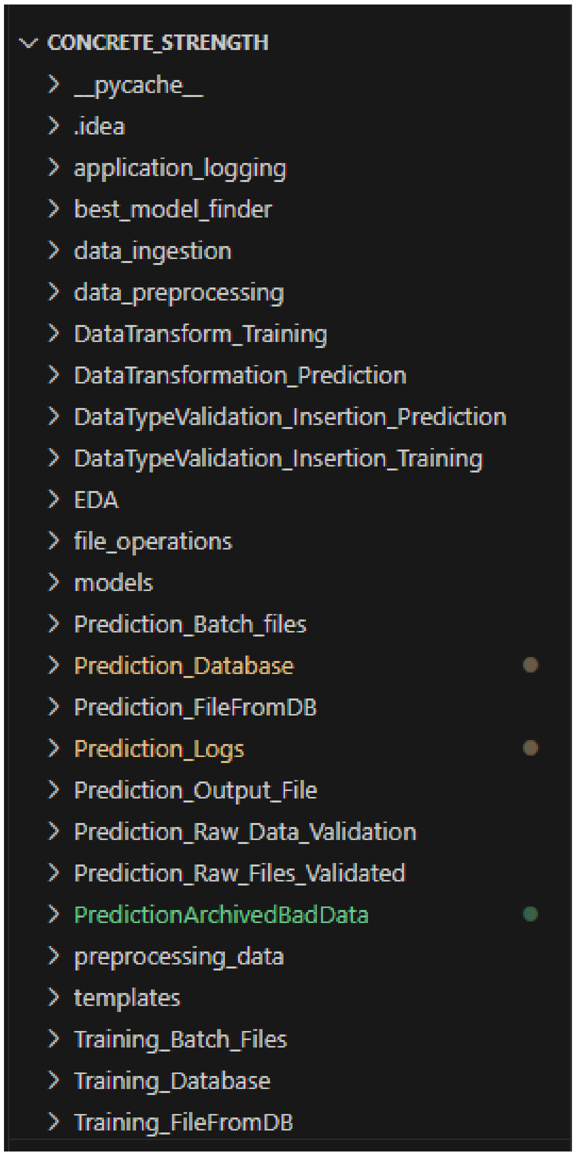
5) Once the prediction is made for all the clusters, the predictions along with the original names before label encoder are saved in a CSV file at a given location and the location is returned to the client.

Deployment

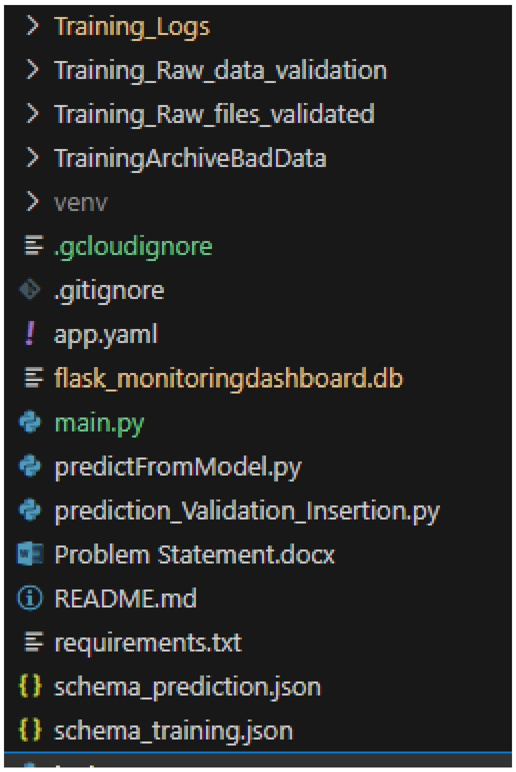
We will be deploying the model to the Google Cloud Platform.

This is a workflow diagram for the prediction of using the trained model.

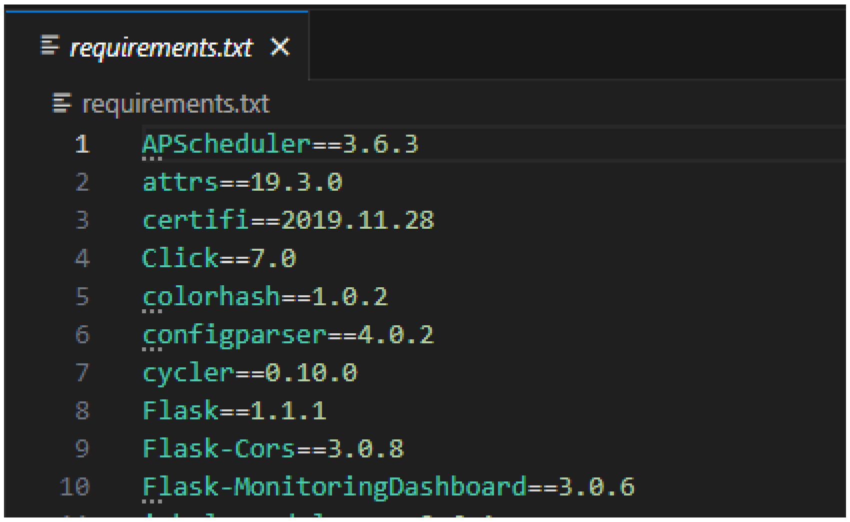
Now let’s see the Concrete_Strength project folder structure.



```
CONCRETE_STRENGTH
  > __pycache__
  > .idea
  > application_logging
  > best_model_finder
  > data_ingestion
  > data_preprocessing
  > DataTransform_Training
  > DataTransformation_Prediction
  > DataTypeValidation_Insertion_Prediction
  > DataTypeValidation_Insertion_Training
  > EDA
  > file_operations
  > models
  > Prediction_Batch_files
  > Prediction_Database
  > Prediction_FileFromDB
  > Prediction_Logs
  > Prediction_Output_File
  > Prediction_Raw_Data_Validation
  > Prediction_Raw_Files_Validated
  > PredictionArchivedBadData
  > preprocessing_data
  > templates
  > Training_Batch_Files
  > Training_Database
  > Training_FileFromDB
```

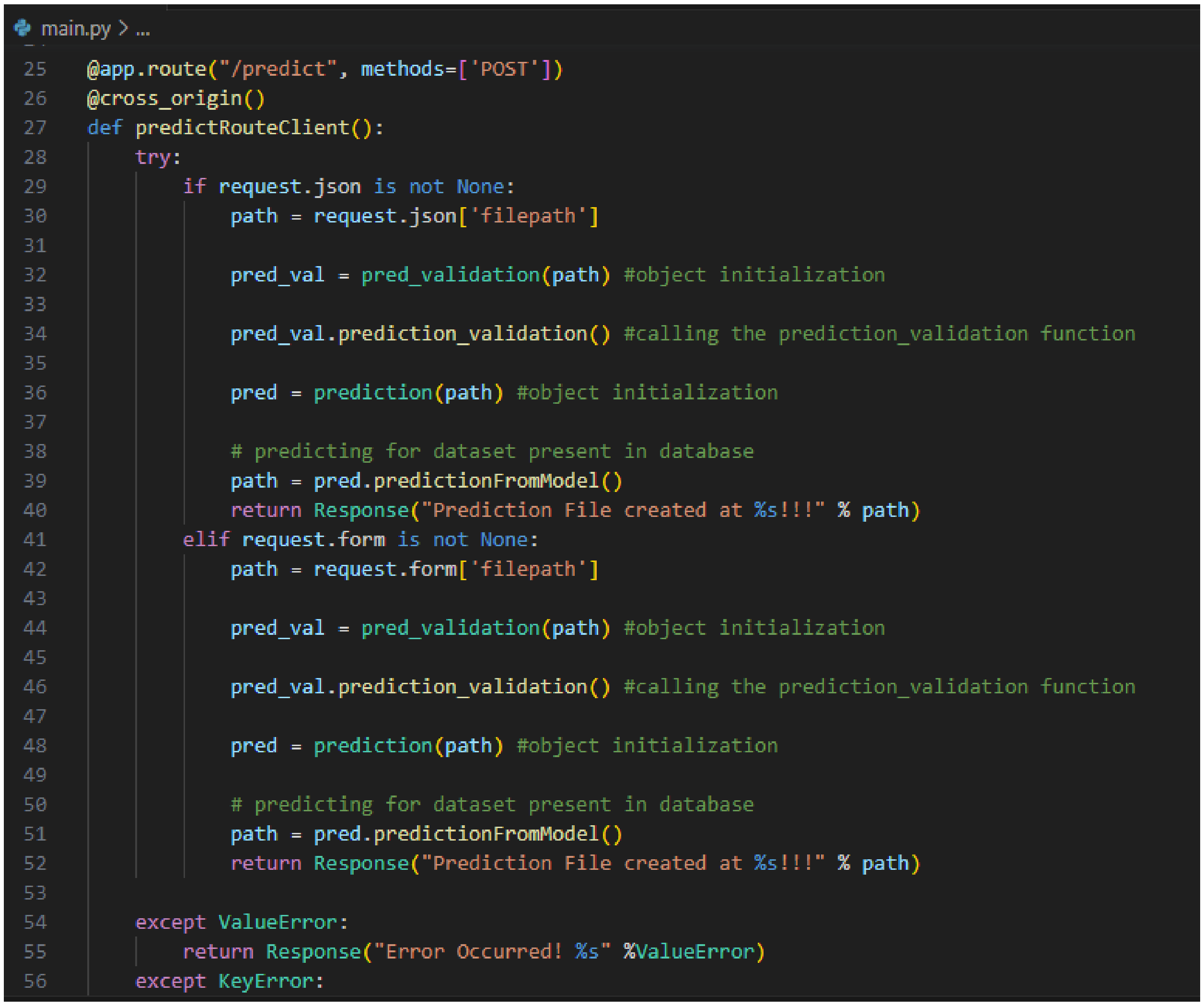


```
> Training_Logs
> Training_Raw_data_validation
> Training_Raw_files_validated
> TrainingArchiveBadData
> venv
.gcloudignore
.gitignore
app.yaml
flask_monitoringdashboard.db
main.py
predictFromModel.py
prediction_Validation_Insertion.py
Problem Statement.docx
README.md
requirements.txt
schema_prediction.json
schema_training.json
```



```
requirements.txt
APScheduler==3.6.3
attrs==19.3.0
certifi==2019.11.28
Click==7.0
colorhash==1.0.2
configparser==4.0.2
cycler==0.10.0
Flask==1.1.1
Flask-Cors==3.0.8
Flask-MonitoringDashboard==3.0.6
```

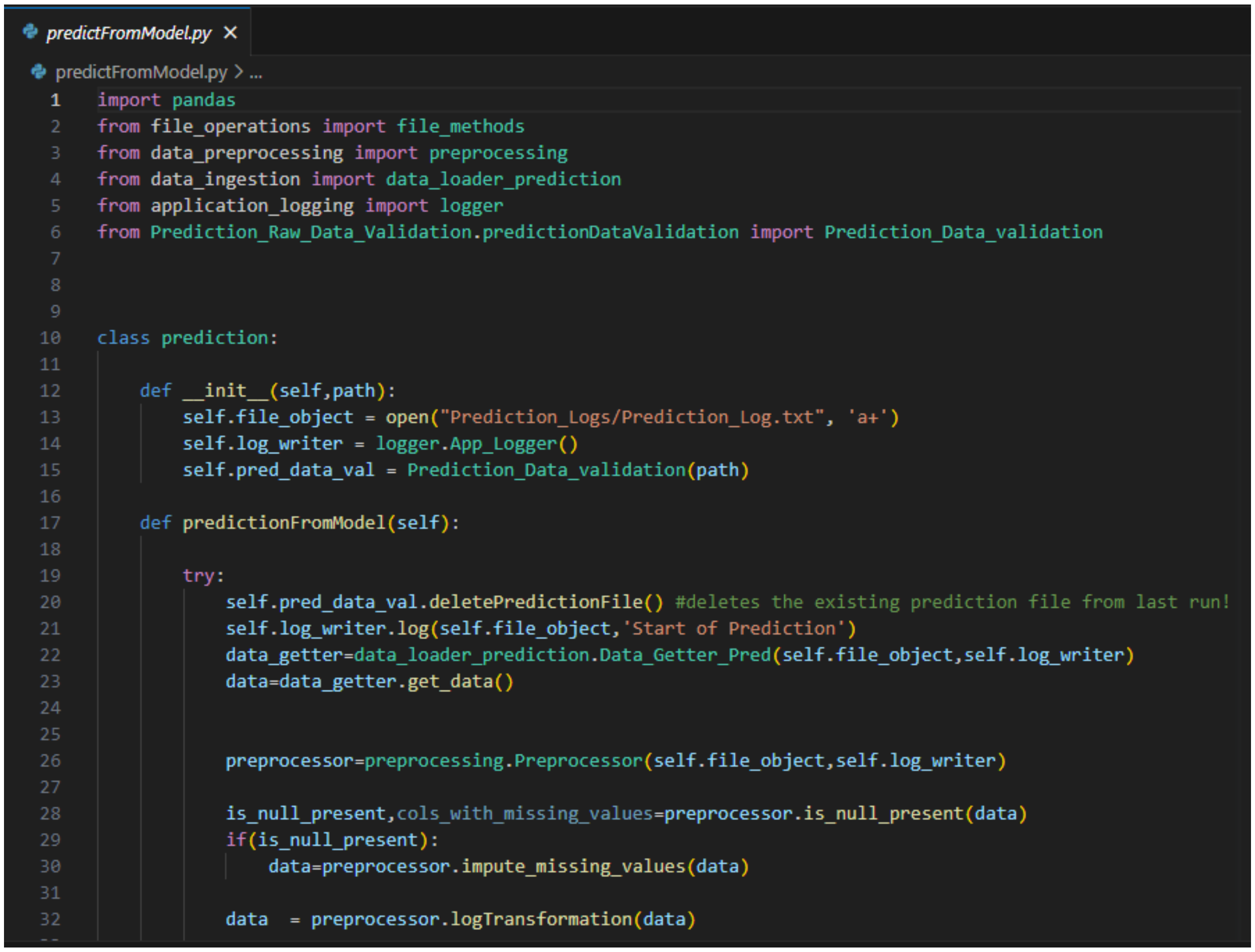
requirements.txt file consists of all the packages that you need to deploy the app in the cloud.



main.py > ...

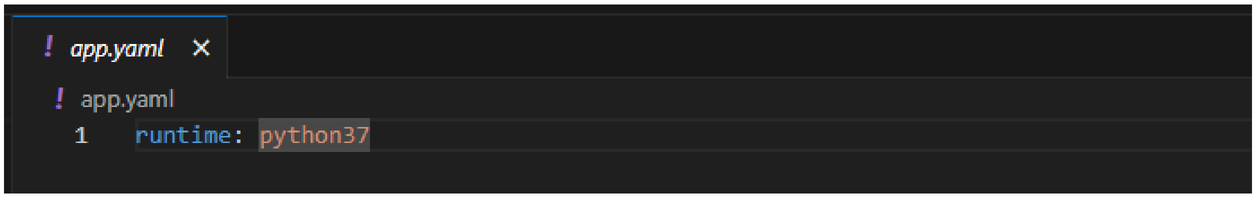
```python
@app.route("/predict", methods=['POST'])
@cross_origin()
def predictRouteClient():
    try:
        if request.json is not None:
            path = request.json['filepath']

            pred_val = pred_validation(path) #object initialization

            pred_val.prediction_validation() #calling the prediction_validation function

            pred = prediction(path) #object initialization

            # predicting for dataset present in database
            path = pred.predictionFromModel()
            return Response("Prediction File created at %s!!!" % path)
        elif request.form is not None:
            path = request.form['filepath']

            pred_val = pred_validation(path) #object initialization

            pred_val.prediction_validation() #calling the prediction_validation function

            pred = prediction(path) #object initialization

            # predicting for dataset present in database
            path = pred.predictionFromModel()
            return Response("Prediction File created at %s!!!" % path)

    except ValueError:
        return Response("Error Occurred! %s" %ValueError)
    except KeyError:
```

main.py is the entry point of our application, where the flask server starts.



```python
import pandas
from file_operations import file_methods
from data_preprocessing import preprocessing
from data_ingestion import data_loader_prediction
from application_logging import logger
from Prediction_Raw_Data_Validation.predictionDataValidation import Prediction_Data_validation



class prediction:

    def __init__(self,path):
        self.file_object = open("Prediction_Logs/Prediction_Log.txt", 'a+')
        self.log_writer = logger.App_Logger()
        self.pred_data_val = Prediction_Data_validation(path)

    def predictionFromModel(self):

        try:
            self.pred_data_val.deletePredictionFile() #deletes the existing prediction file from last run!
            self.log_writer.log(self.file_object,'Start of Prediction')
            data_getter=data_loader_prediction.Data_Getter_Pred(self.file_object,self.log_writer)
            data=data_getter.get_data()


            preprocessor=preprocessing.Preprocessor(self.file_object,self.log_writer)

            is_null_present,cols_with_missing_values=preprocessor.is_null_present(data)
            if(is_null_present):
                data=preprocessor.impute_missing_values(data)

            data  = preprocessor.logTransformation(data)
```

This is the **predictFromModel.py** file where the predictions take place based on the data we are giving as an input to the model.



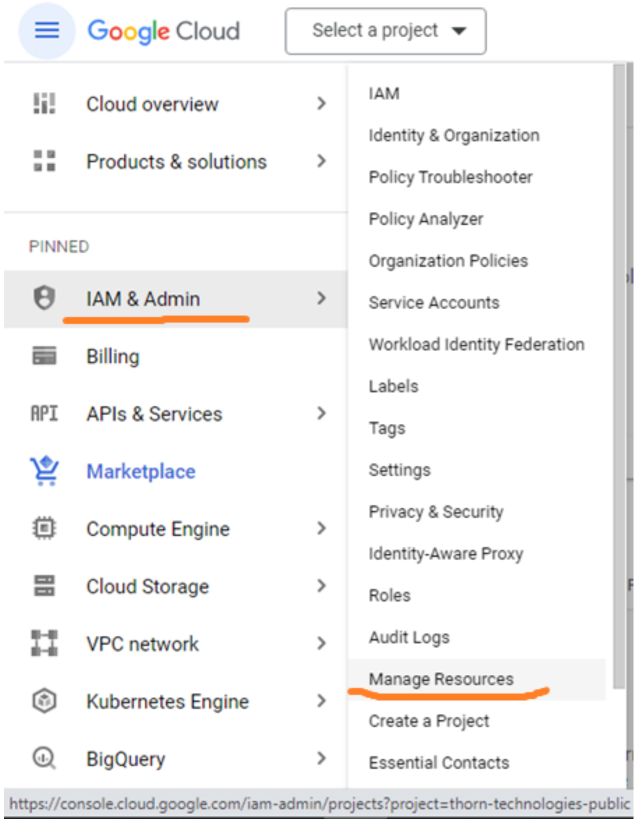
```yaml
runtime: python37
```

app.yaml:- It contains the Python version number.

Deployment to Google cloud (GCP)

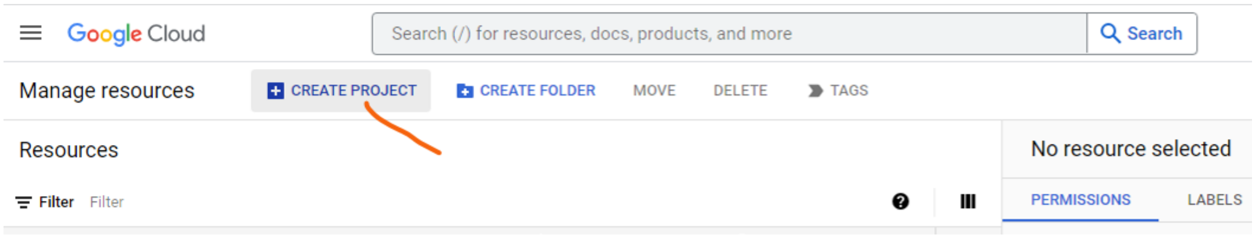
1. Go to https://cloud.google.com/ and create an account if you haven't created one. Then go to the console of your account. https://console.cloud.google.com/

2. Go to IAM & Admin and select Manage Resources

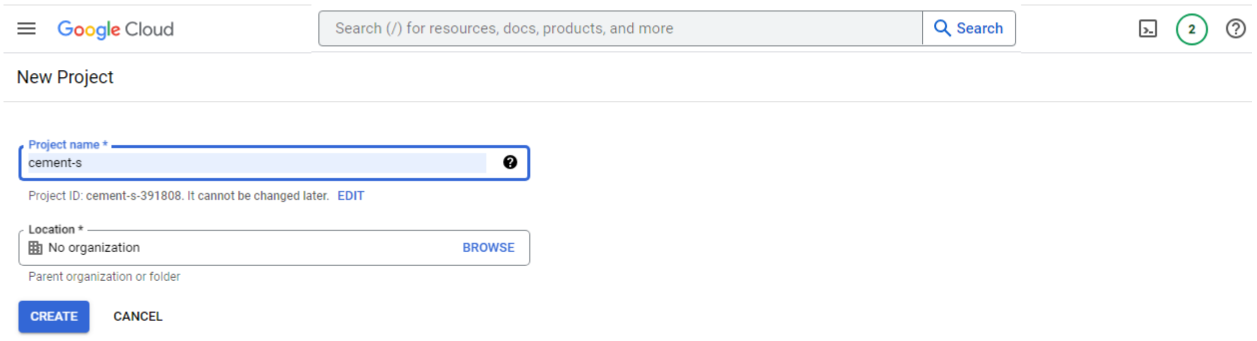


https://console.cloud.google.com/iam-admin/projects?project=thorn-technologies-public

3. In the manage resources page, click on CREATE PROJECT option.



4. Give your project name and Location details. And click on CREATE.



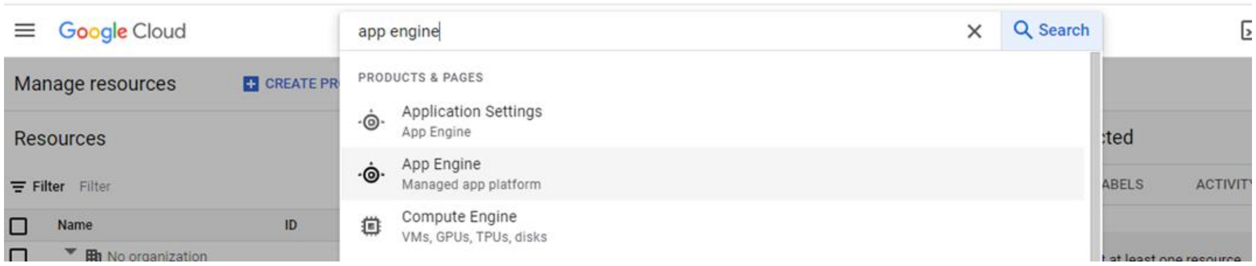
Project name *: cement-s

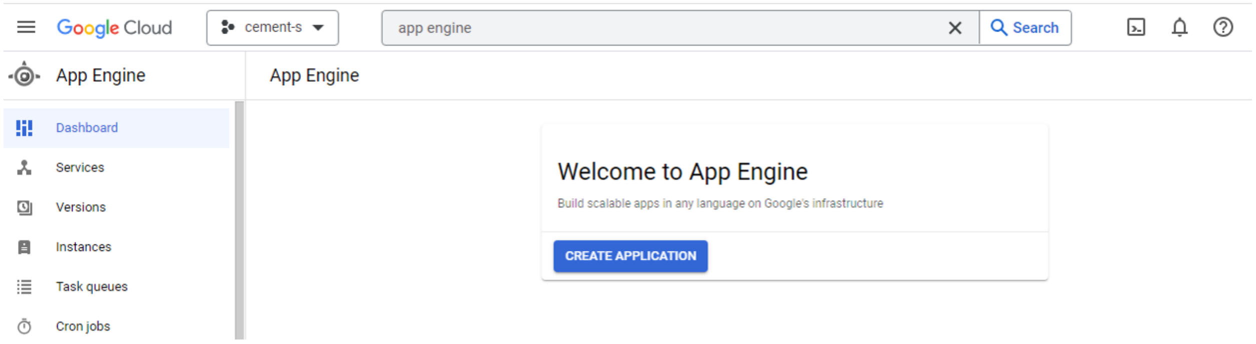
Project ID: cement-s-391808. It cannot be changed later.

Location *: No organization

Parent organization or folder

5. Type app engine in search box and open it.

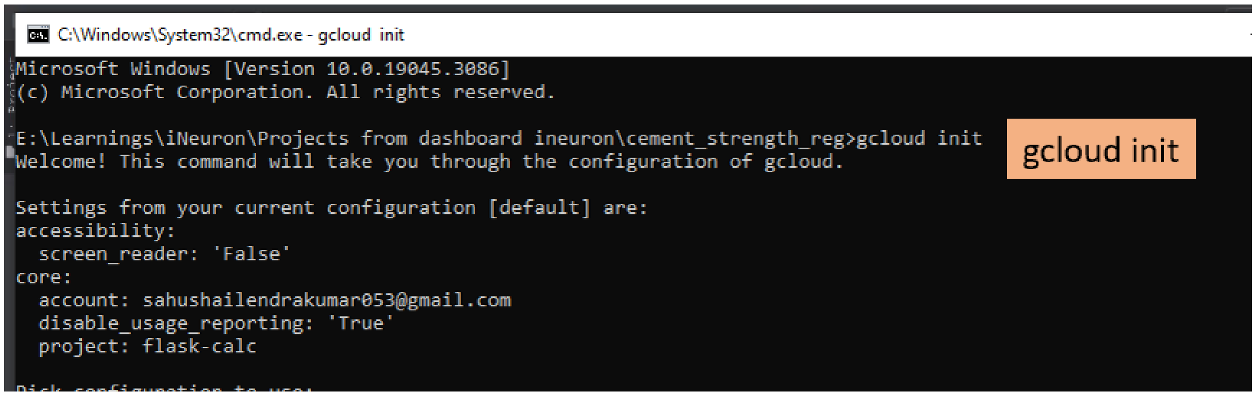




6. Go to https://dl.google.com/dl/cloudsdk/channels/rapid/GoogleCloudSDKInstaller.exe to download the google cloud SDK in your machine. And install it.

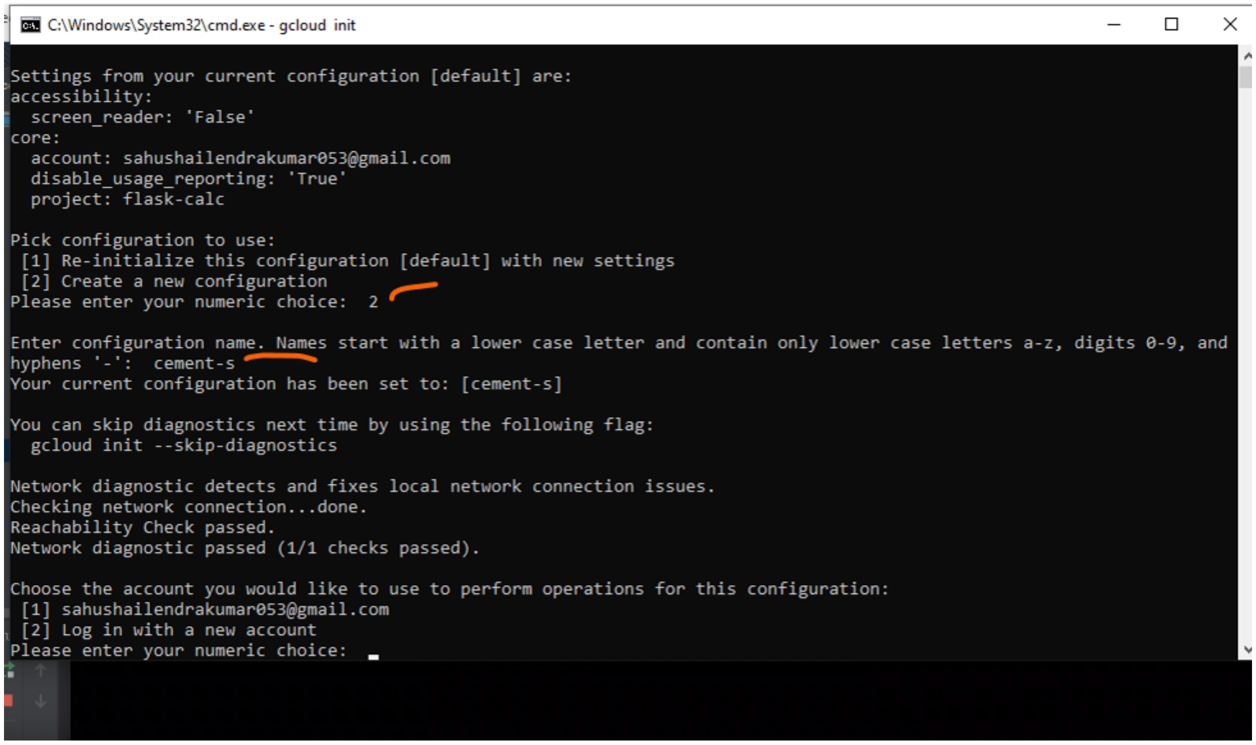
7. Go to the local folder of the project in your machine and launch cmd in that folder.

8. Enter the command gcloud init to initialise the gcloud context.



```
Microsoft Windows [Version 10.0.19045.3086]
(c) Microsoft Corporation. All rights reserved.

E:\Learnings\iNeuron\Projects from dashboard ineuron\cement_strength_reg>gcloud init
Welcome! This command will take you through the configuration of gcloud.

Settings from your current configuration [default] are:
accessibility:
  screen_reader: 'False'
core:
  account: sahushailendrakumar053@gmail.com
  disable_usage_reporting: 'True'
  project: flask-calc
```

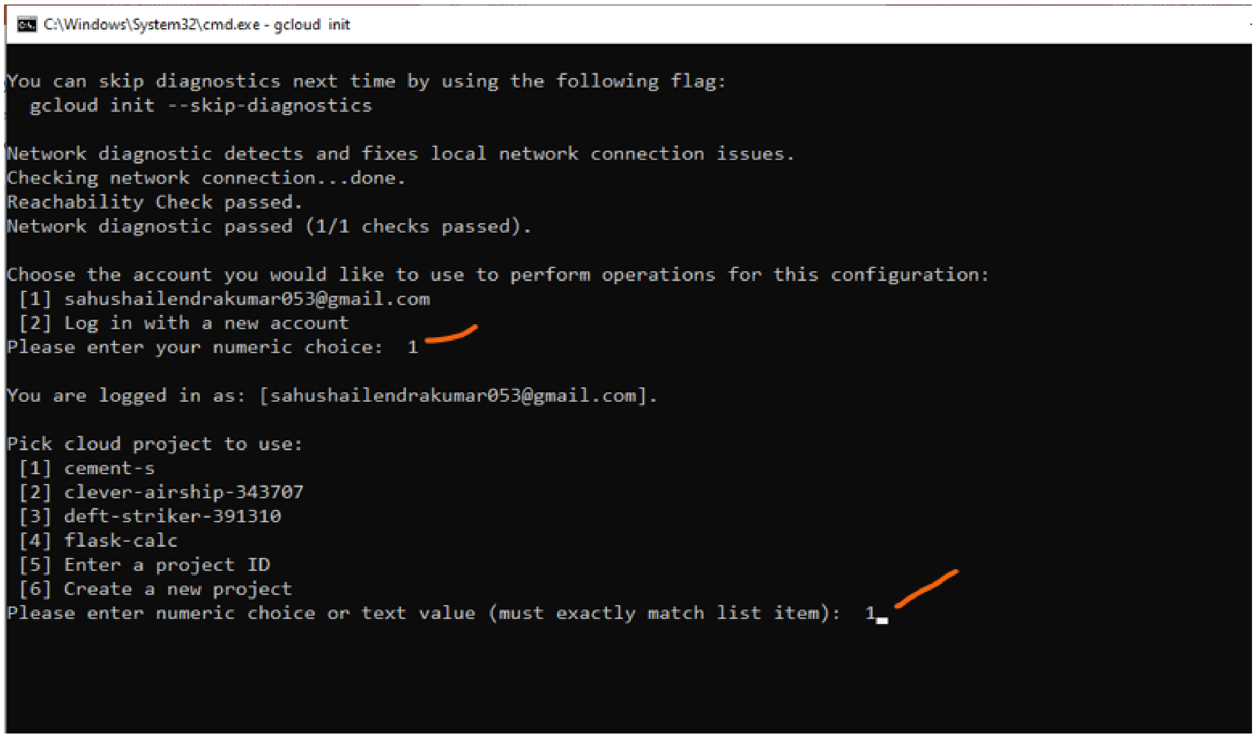
9. Select option 2 for new configuration and then write the name of the project that you created previously.



```
C:\Windows\System32\cmd.exe - gcloud init

Settings from your current configuration [default] are:
accessibility:
  screen_reader: 'False'
core:
  account: sahushailendrakumar053@gmail.com
  disable_usage_reporting: 'True'
  project: flask-calc

Pick configuration to use:
 [1] Re-initialize this configuration [default] with new settings
 [2] Create a new configuration
Please enter your numeric choice:  2

Enter configuration name. Names start with a lower case letter and contain only lower case letters a-z, digits 0-9, and
hyphens '-':  cement-s
Your current configuration has been set to: [cement-s]

You can skip diagnostics next time by using the following flag:
  gcloud init --skip-diagnostics

Network diagnostic detects and fixes local network connection issues.
Checking network connection...done.
Reachability Check passed.
Network diagnostic passed (1/1 checks passed).

Choose the account you would like to use to perform operations for this configuration:
 [1] sahushailendrakumar053@gmail.com
 [2] Log in with a new account
Please enter your numeric choice:  _
```

10. Select the account that you want to use or login with another.

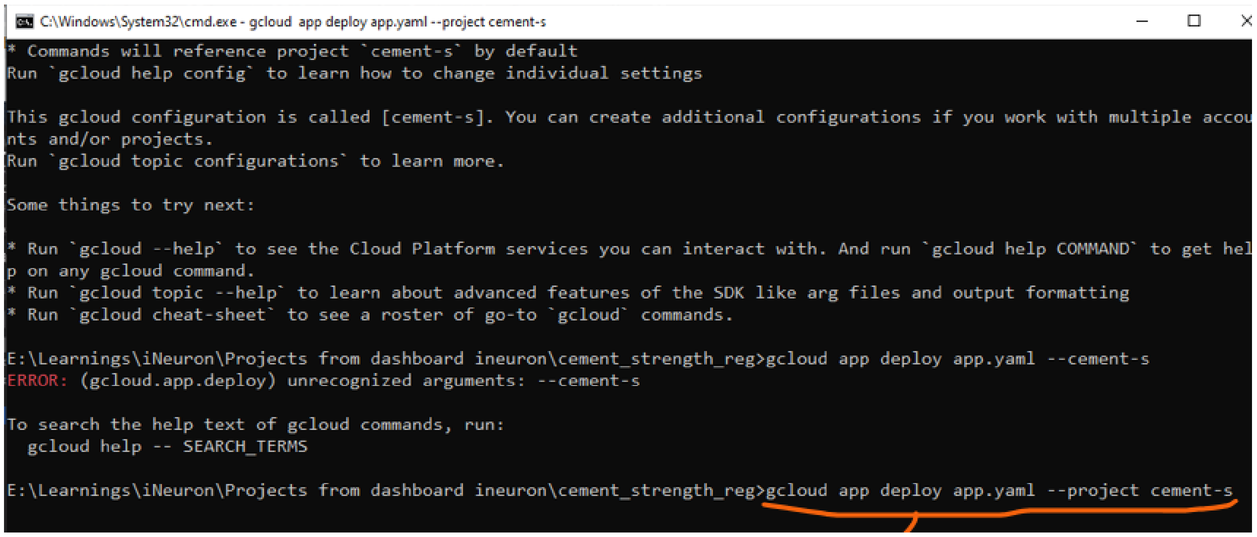
    Then pick the project name to use.



```
C:\Windows\System32\cmd.exe - gcloud init

You can skip diagnostics next time by using the following flag:
  gcloud init --skip-diagnostics

Network diagnostic detects and fixes local network connection issues.
Checking network connection...done.
Reachability Check passed.
Network diagnostic passed (1/1 checks passed).

Choose the account you would like to use to perform operations for this configuration:
 [1] sahushailendrakumar053@gmail.com
 [2] Log in with a new account
Please enter your numeric choice:  1

You are logged in as: [sahushailendrakumar053@gmail.com].

Pick cloud project to use:
 [1] cement-s
 [2] clever-airship-343707
 [3] deft-striker-391310
 [4] flask-calc
 [5] Enter a project ID
 [6] Create a new project
Please enter numeric choice or text value (must exactly match list item):  1_
```

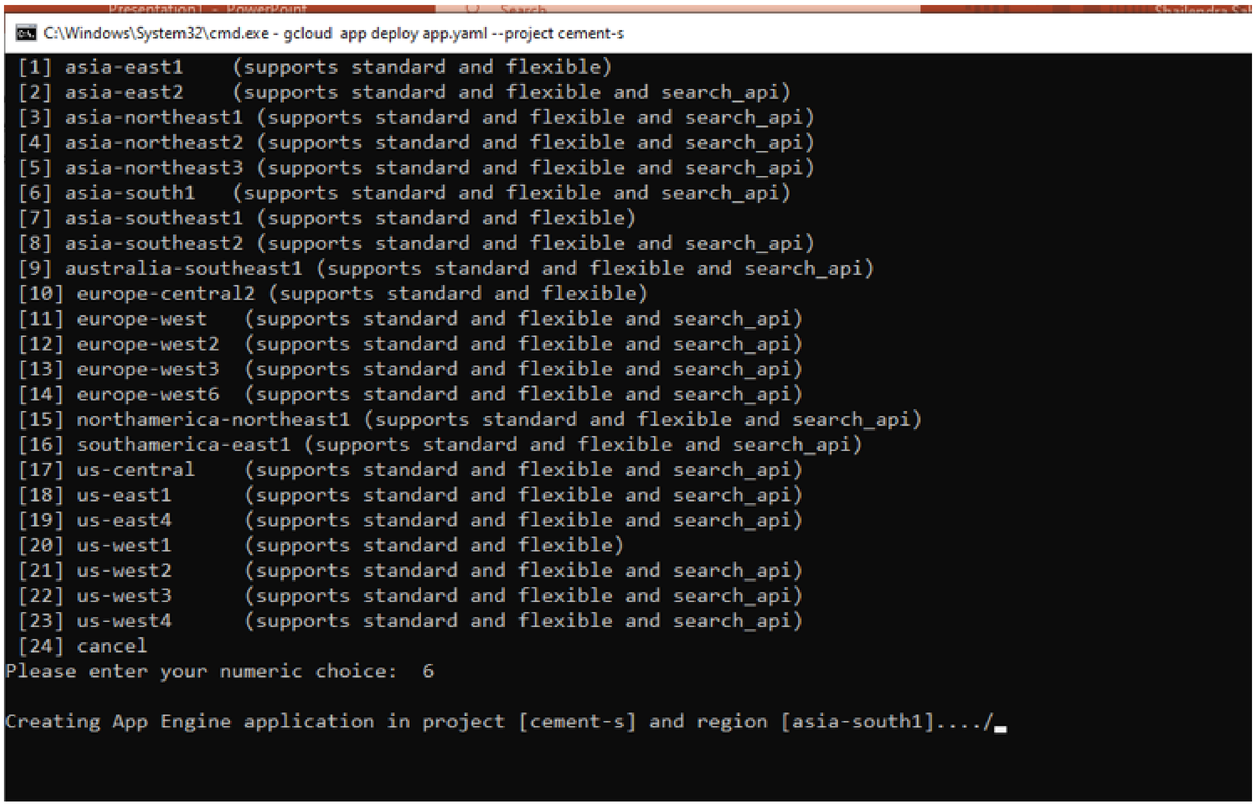
11. Run this command to deploy the project

    gcloud app deploy app.yaml --project <<project name>>



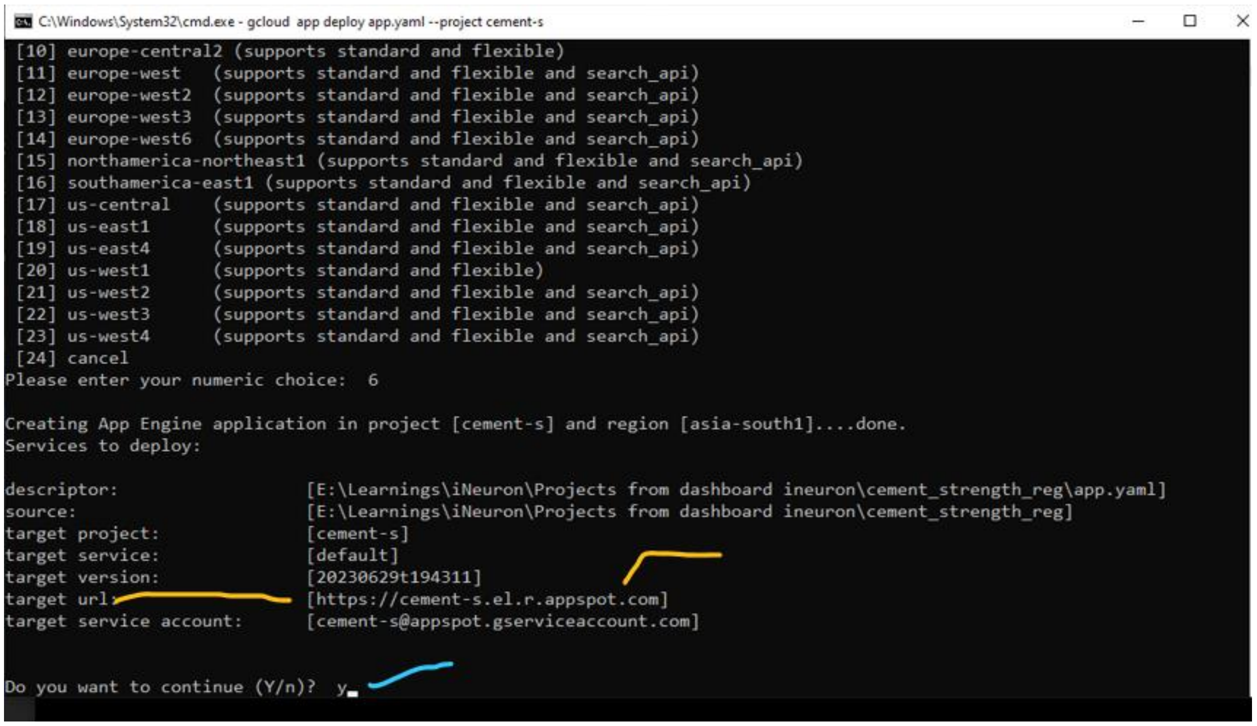
```
C:\Windows\System32\cmd.exe - gcloud  app deploy app.yaml --project cement-s
* Commands will reference project `cement-s` by default
Run `gcloud help config` to learn how to change individual settings

This gcloud configuration is called [cement-s]. You can create additional configurations if you work with multiple accou
nts and/or projects.
Run `gcloud topic configurations` to learn more.

Some things to try next:

* Run `gcloud --help` to see the Cloud Platform services you can interact with. And run `gcloud help COMMAND` to get hel
p on any gcloud command.
* Run `gcloud topic --help` to learn about advanced features of the SDK like arg files and output formatting
* Run `gcloud cheat-sheet` to see a roster of go-to `gcloud` commands.

E:\Learnings\iNeuron\Projects from dashboard ineuron\cement_strength_reg>gcloud app deploy app.yaml --cement-s
ERROR: (gcloud.app.deploy) unrecognized arguments: --cement-s

To search the help text of gcloud commands, run:
  gcloud help -- SEARCH_TERMS

E:\Learnings\iNeuron\Projects from dashboard ineuron\cement_strength_reg>gcloud app deploy app.yaml --project cement-s
```

12. Now choose your region.



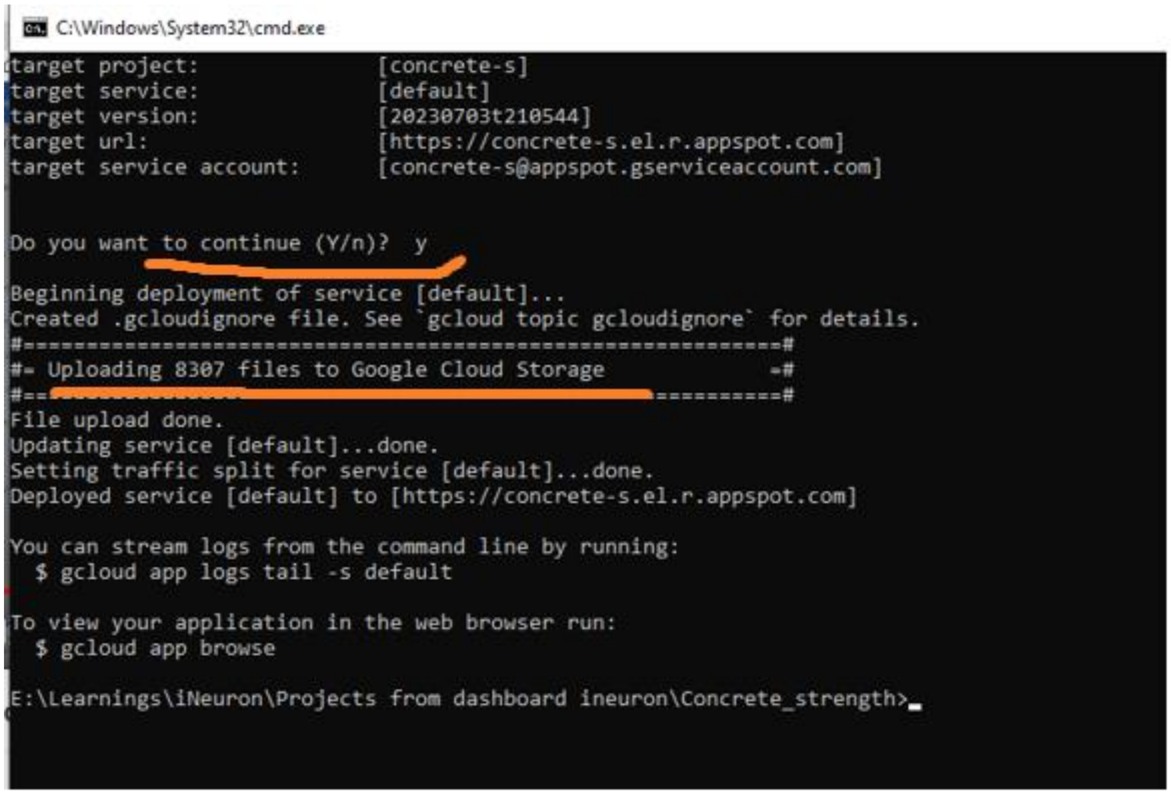
```
C:\Windows\System32\cmd.exe - gcloud  app deploy app.yaml --project cement-s
 [1] asia-east1     (supports standard and flexible)
 [2] asia-east2     (supports standard and flexible and search_api)
 [3] asia-northeast1 (supports standard and flexible and search_api)
 [4] asia-northeast2 (supports standard and flexible and search_api)
 [5] asia-northeast3 (supports standard and flexible and search_api)
 [6] asia-south1    (supports standard and flexible and search_api)
 [7] asia-southeast1 (supports standard and flexible)
 [8] asia-southeast2 (supports standard and flexible and search_api)
 [9] australia-southeast1 (supports standard and flexible and search_api)
 [10] europe-central2 (supports standard and flexible)
 [11] europe-west     (supports standard and flexible and search_api)
 [12] europe-west2    (supports standard and flexible and search_api)
 [13] europe-west3    (supports standard and flexible and search_api)
 [14] europe-west6    (supports standard and flexible and search_api)
 [15] northamerica-northeast1 (supports standard and flexible and search_api)
 [16] southamerica-east1 (supports standard and flexible and search_api)
 [17] us-central      (supports standard and flexible and search_api)
 [18] us-east1        (supports standard and flexible and search_api)
 [19] us-east4        (supports standard and flexible and search_api)
 [20] us-west1        (supports standard and flexible)
 [21] us-west2        (supports standard and flexible and search_api)
 [22] us-west3        (supports standard and flexible and search_api)
 [23] us-west4        (supports standard and flexible and search_api)
 [24] cancel
Please enter your numeric choice:  6

Creating App Engine application in project [cement-s] and region [asia-south1]..../
```

13. It will complete the creation of app engine application. And will give us the target url to access the prediction page.



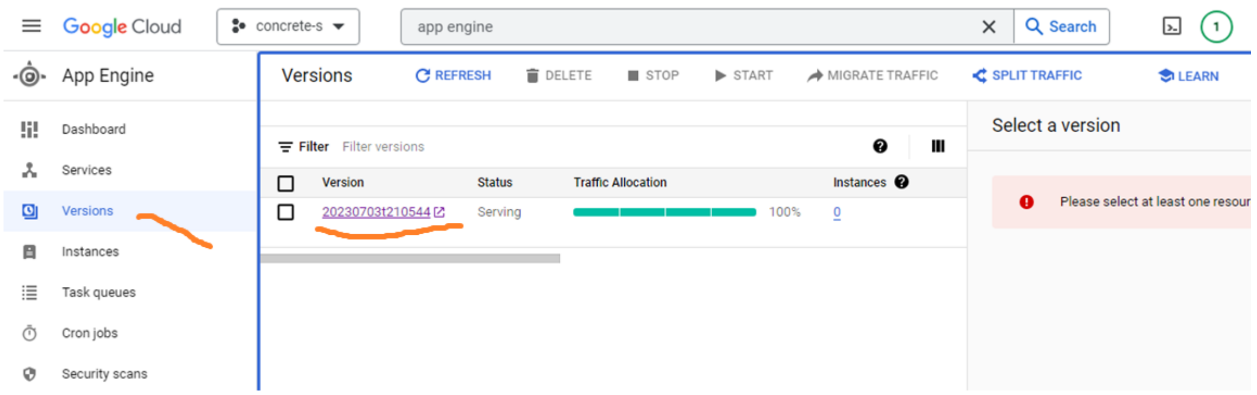
```
C:\Windows\System32\cmd.exe - gcloud  app deploy app.yaml --project cement-s
 [10] europe-central2 (supports standard and flexible)
 [11] europe-west   (supports standard and flexible and search_api)
 [12] europe-west2  (supports standard and flexible and search_api)
 [13] europe-west3  (supports standard and flexible and search_api)
 [14] europe-west6  (supports standard and flexible and search_api)
 [15] northamerica-northeast1 (supports standard and flexible and search_api)
 [16] southamerica-east1 (supports standard and flexible and search_api)
 [17] us-central    (supports standard and flexible and search_api)
 [18] us-east1      (supports standard and flexible and search_api)
 [19] us-east4      (supports standard and flexible and search_api)
 [20] us-west1      (supports standard and flexible)
 [21] us-west2      (supports standard and flexible and search_api)
 [22] us-west3      (supports standard and flexible and search_api)
 [23] us-west4      (supports standard and flexible and search_api)
 [24] cancel
Please enter your numeric choice:  6

Creating App Engine application in project [cement-s] and region [asia-south1]....done.
Services to deploy:

descriptor:                  [E:\Learnings\iNeuron\Projects from dashboard ineuron\cement_strength_reg\app.yaml]
source:                      [E:\Learnings\iNeuron\Projects from dashboard ineuron\cement_strength_reg]
target project:              [cement-s]
target service:              [default]
target version:              [20230629t194311]
target url:                  [https://cement-s.el.r.appspot.com]
target service account:      [cement-s@appspot.gserviceaccount.com]


Do you want to continue (Y/n)?  y
```

14. It will ask to continue uploading all the files into cloud.
Select Y to give it yes to continue.



```
C:\Windows\System32\cmd.exe
target project:              [concrete-s]
target service:              [default]
target version:              [20230703t210544]
target url:                  [https://concrete-s.el.r.appspot.com]
target service account:      [concrete-s@appspot.gserviceaccount.com]


Do you want to continue (Y/n)?  y

Beginning deployment of service [default]...
Created .gcloudignore file. See `gcloud topic gcloudignore` for details.
#============================================================#
#= Uploading 8307 files to Google Cloud Storage             =#
#============================================================#
File upload done.
Updating service [default]...done.
Setting traffic split for service [default]...done.
Deployed service [default] to [https://concrete-s.el.r.appspot.com]

You can stream logs from the command line by running:
  $ gcloud app logs tail -s default

To view your application in the web browser run:
  $ gcloud app browse

E:\Learnings\iNeuron\Projects from dashboard ineuron\Concrete_strength>
```

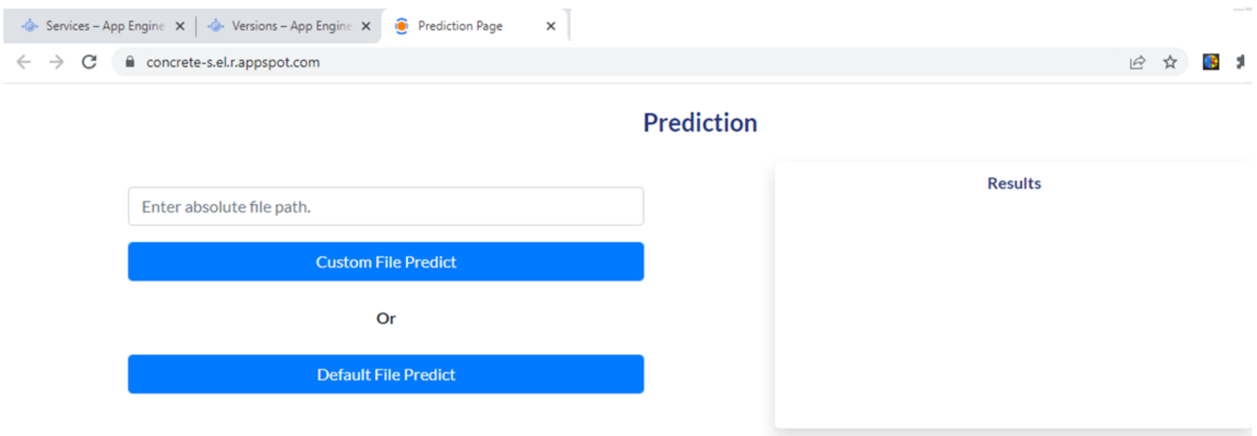
15. After the upload is finished, copy the target url and paste it in the browser.
    Or we can also go to the app engine then version and click on this link to open it in the browser.



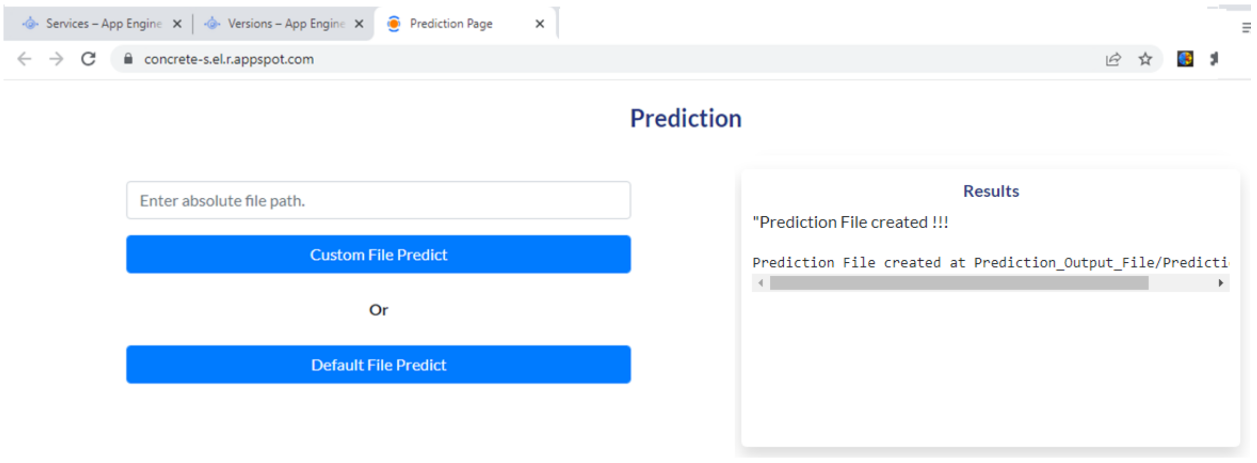
Version	Status	Traffic Allocation	Instances
20230703t210544	Serving	100%	0

Please select at least one resour

16. The prediction page will open. It has two options, custom file predict and default file predict. Select default file predict to make predictions for the data kept in a default folder.



17. The prediction is completed, and prediction file is created.



"Prediction File created !!!

```
Prediction File created at Prediction_Output_File/Predicti
```