**Problem Statement**

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

**Architecture**



**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](https://en.wikipedia.org/wiki/Coal_combustion_product) that is composed of the [particulates](https://en.wikipedia.org/wiki/Particulates) (fine particles of burned fuel) that are driven out of coal-fired [boilers](https://en.wikipedia.org/wiki/Boiler) together with the [flue gases](https://en.wikipedia.org/wiki/Flue_gas). |
| 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](https://en.wikipedia.org/wiki/Concrete) or [mortar](https://en.wikipedia.org/wiki/Mortar_(masonry)) allows the reduction of the water to cement ratio without negatively affecting the [workability](https://en.wikipedia.org/wiki/Workability) of the mixture, and enables the production of [self-consolidating concrete](https://en.wikipedia.org/wiki/Self-consolidating_concrete) and high performance concrete |
| Coarse Aggregate (component 6) | quantitative | kg in a m3 mixture | Input Variable-- construction aggregate, or simply "[aggregate](https://en.wikipedia.org/wiki/Aggregate_(composite))", is a broad category of coarse to medium grained particulate material used in [construction](https://en.wikipedia.org/wiki/Construction), including [sand](https://en.wikipedia.org/wiki/Sand), [gravel](https://en.wikipedia.org/wiki/Gravel), [crushed stone](https://en.wikipedia.org/wiki/Crushed_stone), [slag](https://en.wikipedia.org/wiki/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.

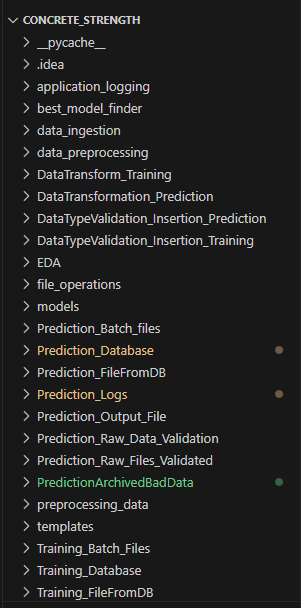
5) Once the prediction is made for all the clusters, the predictions along with the original names before label encoder are saved in a CSV file at a given location and the location is returned to the client.

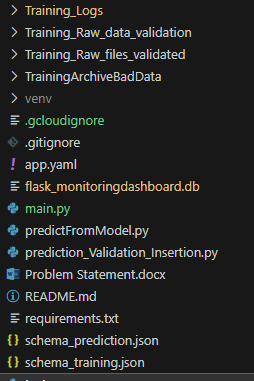
**Deployment**

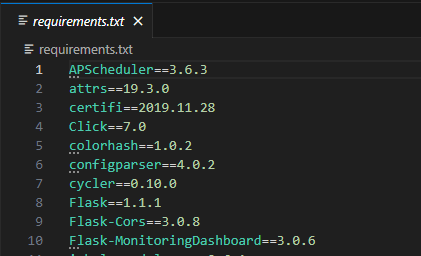
We will be deploying the model to the Google Cloud Platform.

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

**Now let’s see the Concrete\_Strength project folder structure.**



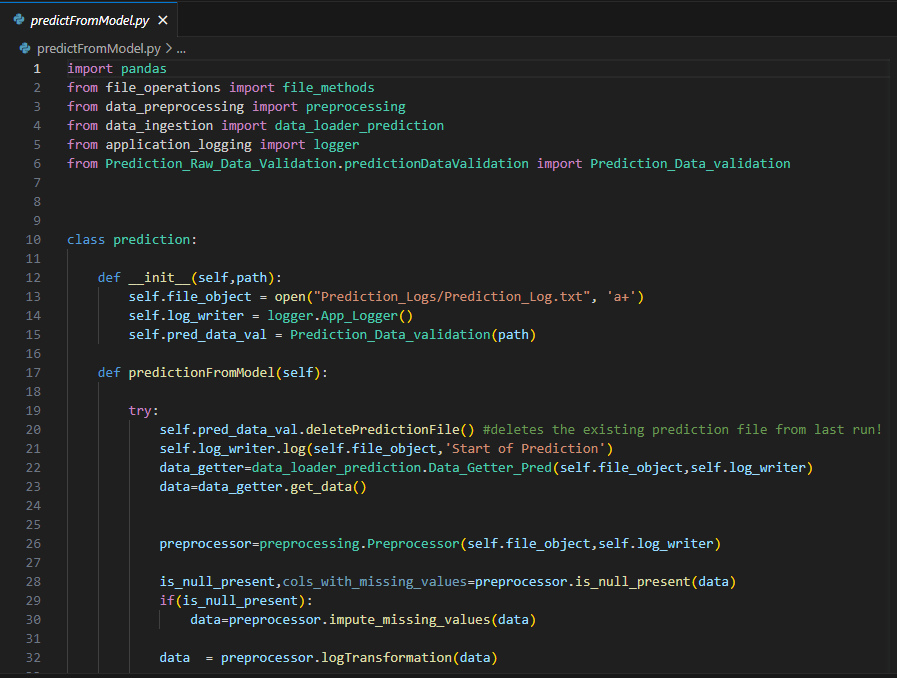




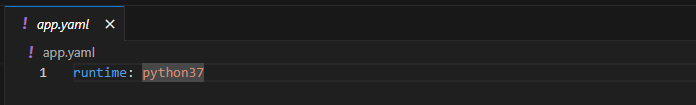
**requirements.txt** file consists of all the packages that you need to deploy the app in the cloud.



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



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



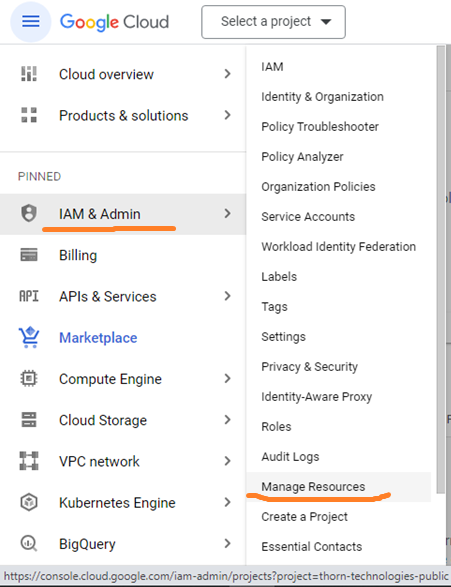
app.yaml:- It contains the Python version number.

**Deployment to Google cloud (GCP)**

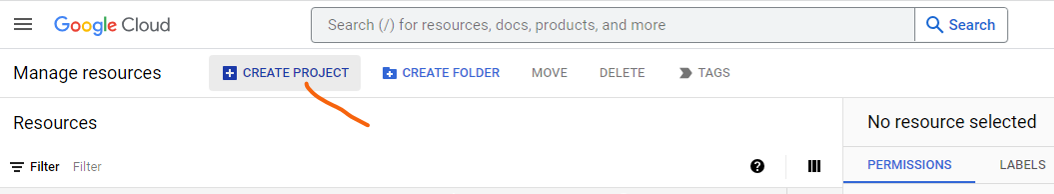
1. Go to <https://cloud.google.com/> and create an account if you haven’t created one.

Then go to the console of your account. <https://console.cloud.google.com/>

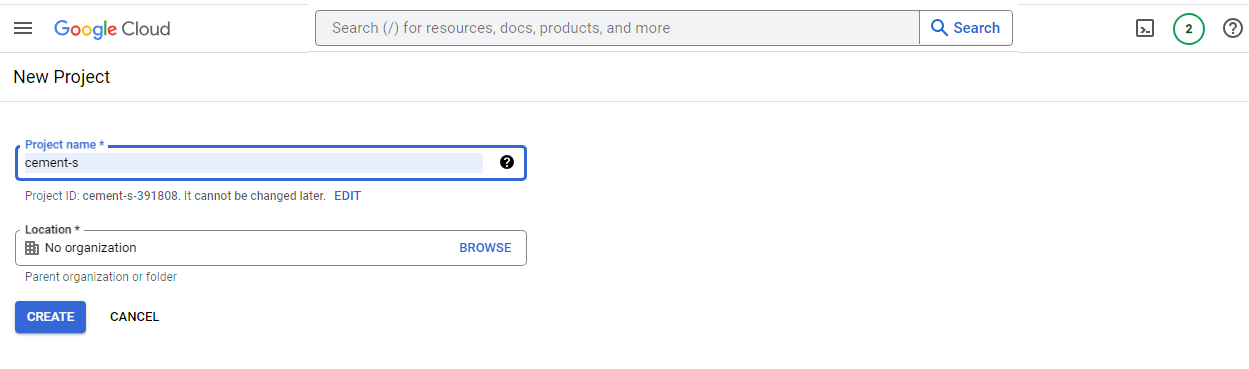
1. Go to IAM & Admin and select Manage Resources



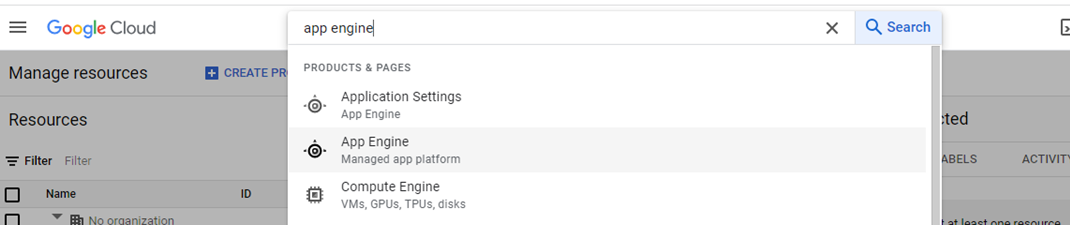
1. In the manage resources page, click on CREATE PROJECT option.

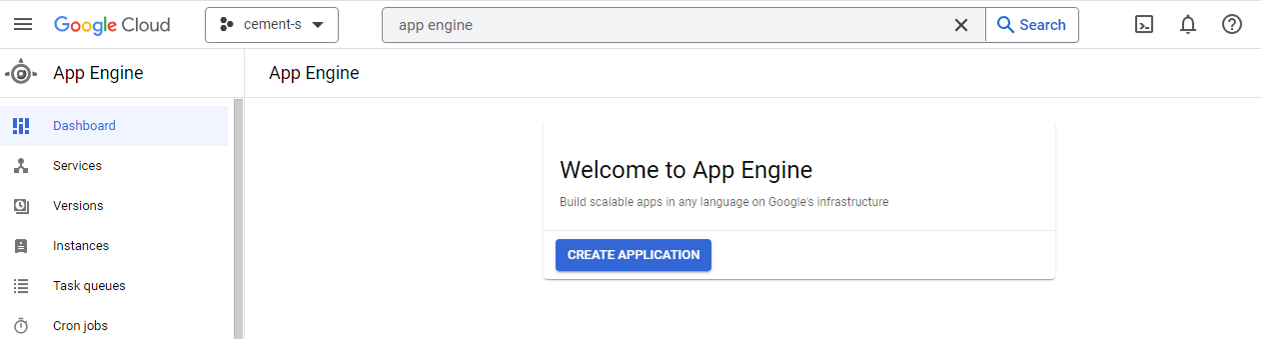


1. Give your project name and Location details. And click on CREATE.

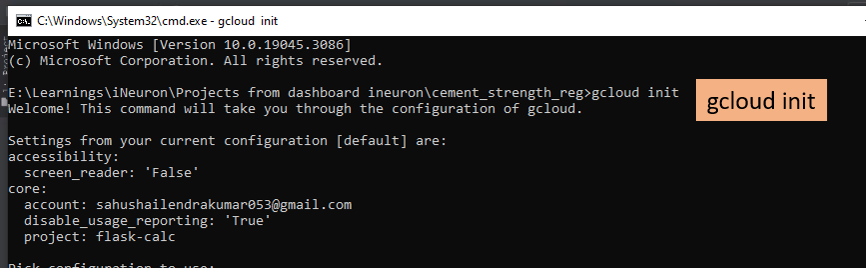


1. Type app engine in search box and open it.

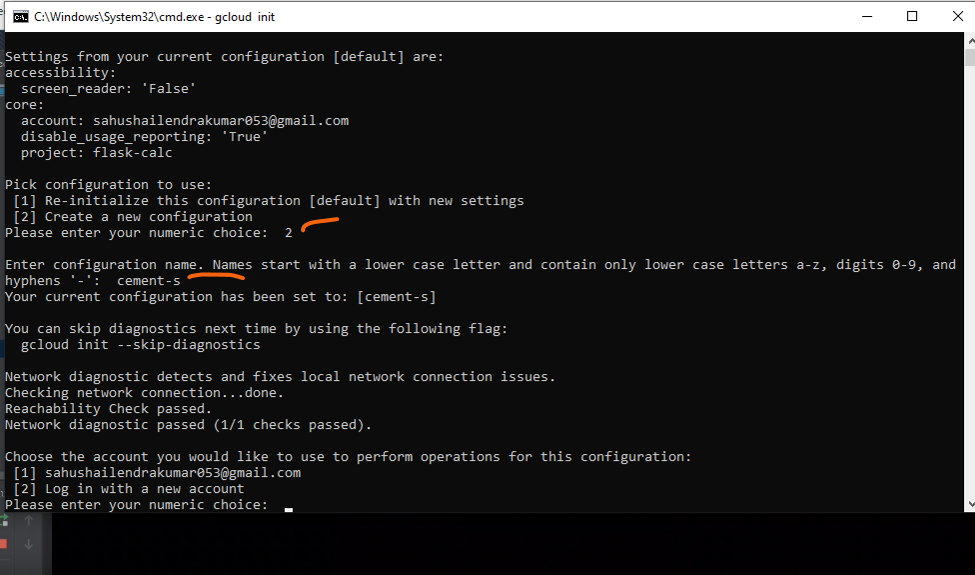




1. Go to <https://dl.google.com/dl/cloudsdk/channels/rapid/GoogleCloudSDKInstaller.exe> to download the google cloud SDK in your machine. And install it.
2. Go to the local folder of the project in your machine and launch cmd in that folder.
3. Enter the command gcloud init to initialise the gcloud context.

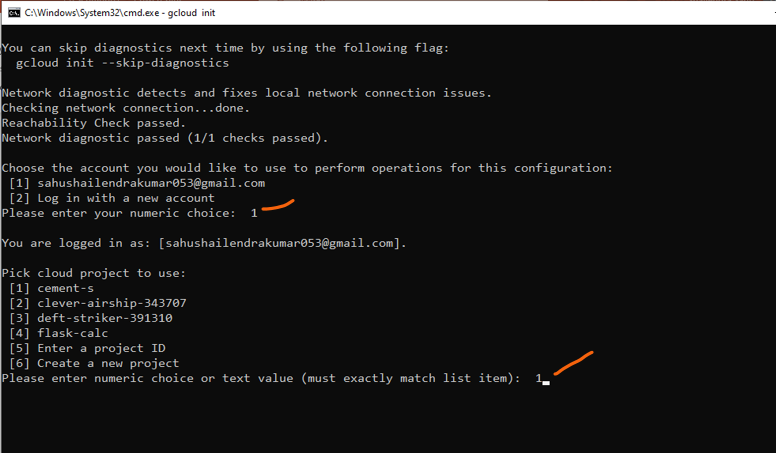


1. Select option 2 for new configuration and then write the name of the project that you created previously.



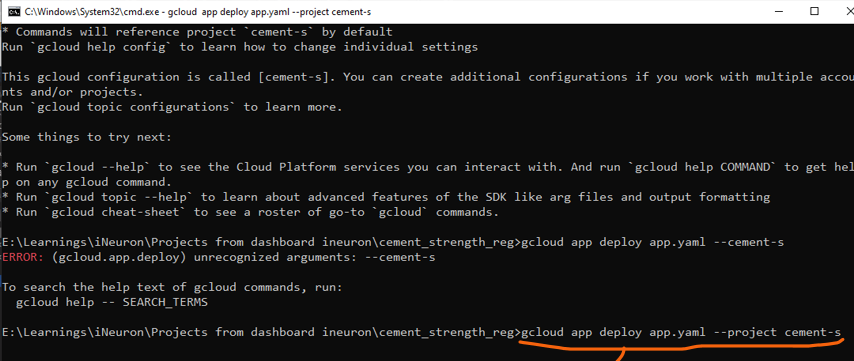
1. Select the account that you want to use or login with another.

Then pick the project name to use.

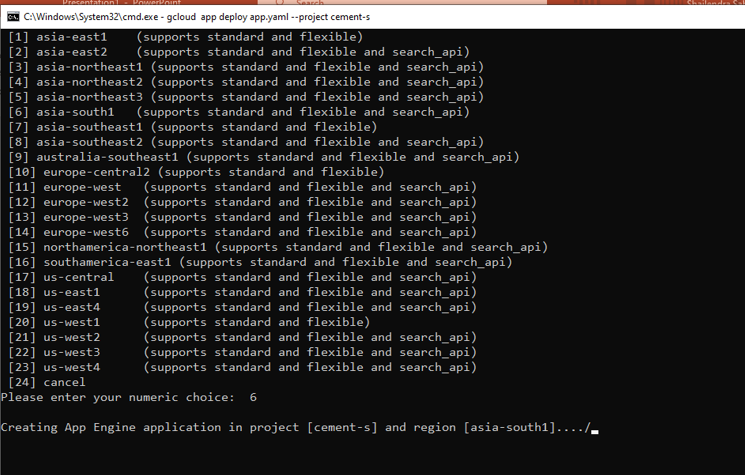


1. Run this command to deploy the project

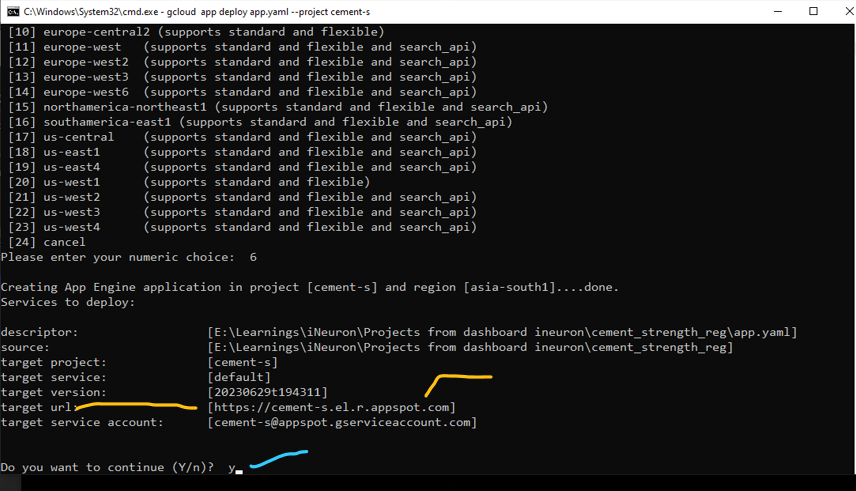
gcloud app deploy app.yaml --project <<project name>>



1. Now choose your region.

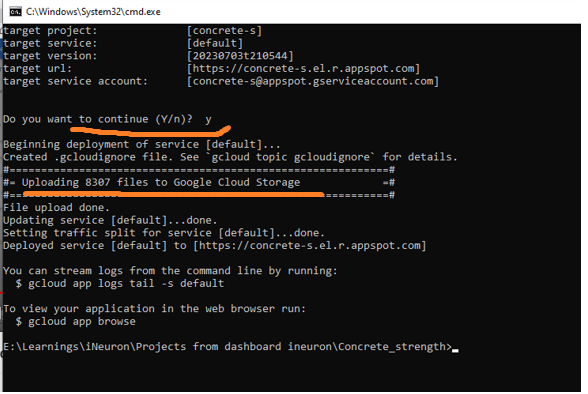


1. It will complete the creation of app engine application. And will give us the target url to access the prediction page.



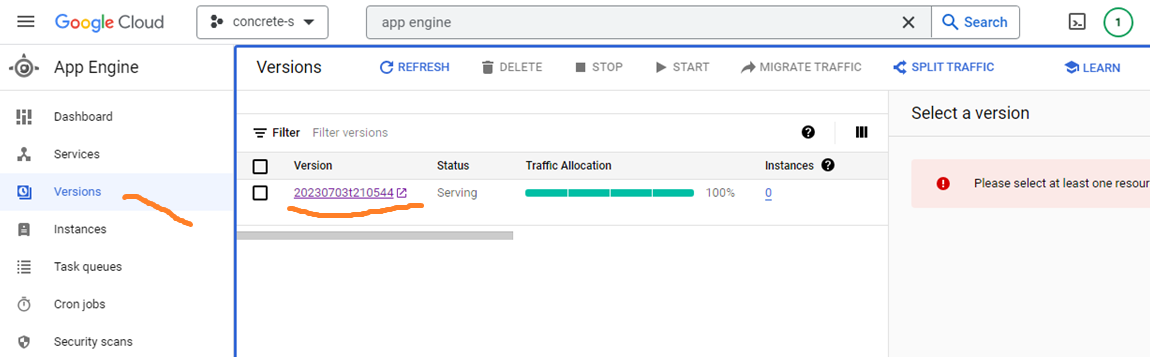
1. It will ask to continue uploading all the files into cloud.

Select Y to give it yes to continue.



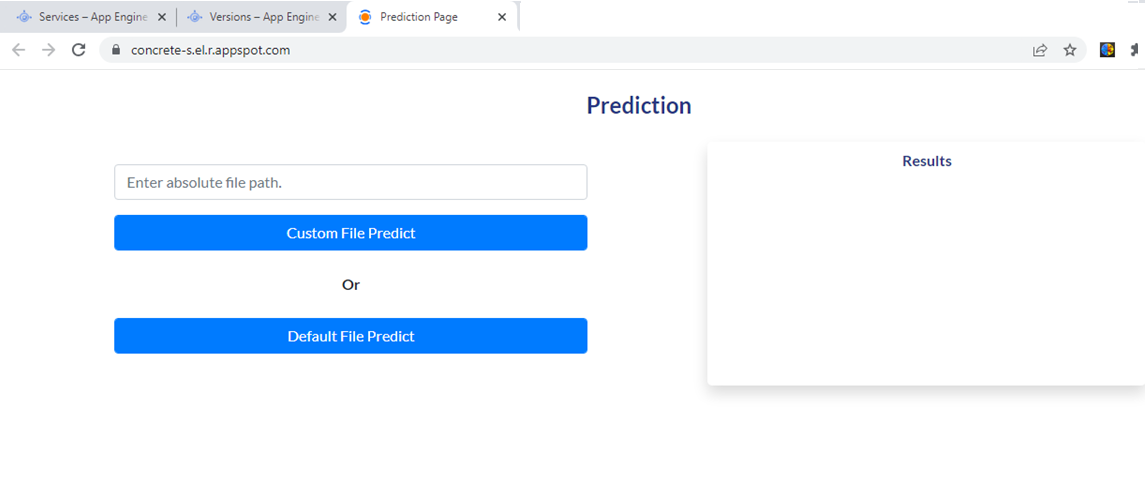
1. After the upload is finished, copy the target url and paste it in the browser.

Or we can also go to the app engine then version and click on this link to open it in the browser.



1. 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.



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