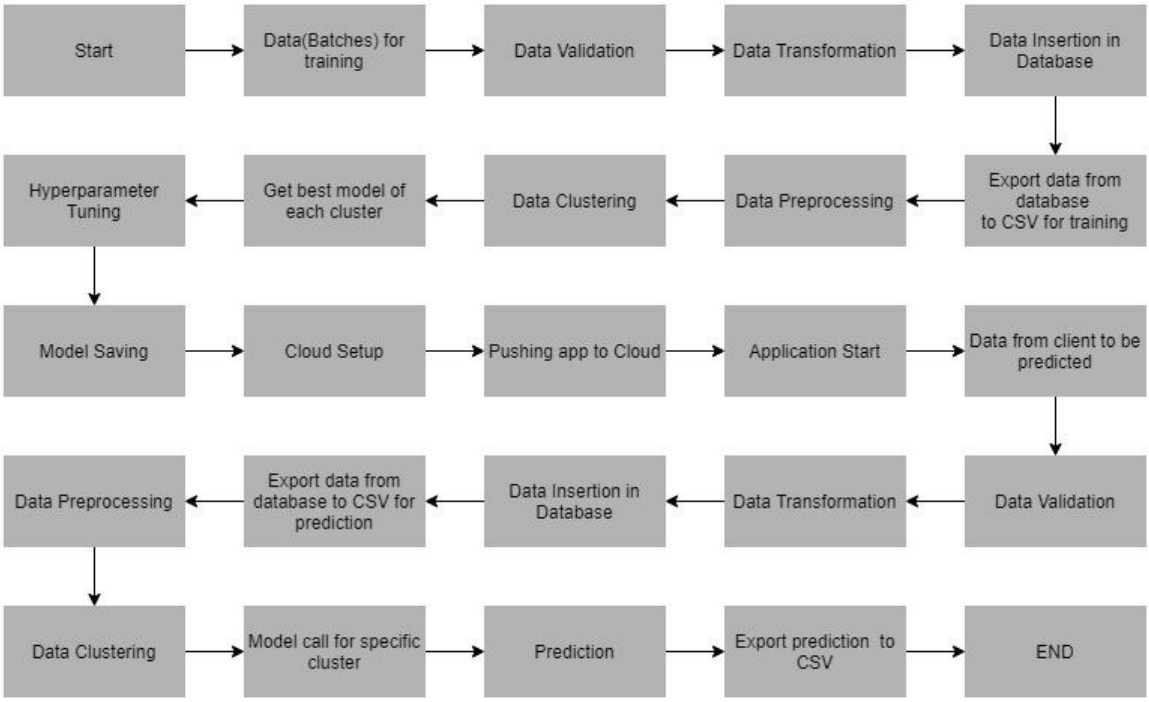
Problem Statement

To build a regression model to predict the heating load and cooling load 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 "EcoComfort: Intelligent Energy Efficiency" 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
Relative Compactness	quantitative	Fraction Between (0-1)	Input Variable - Relative compactness is a dimensionless quantity and does not have a specific unit. It

			is a ratio or a coefficient comparing the actual surface area of the building to the surface area of a hypothetical cuboid with the same volume.
Surface Area	quantitative	square meters (m^2)	Input Variable - It represents the total area of the building envelope, including walls, roof, windows, and other external surfaces.
Wall Area	quantitative	square meters (m^2)	Input Variable- refers to the total surface area of the walls of the building.
Roof Area	quantitative	square meters (m^2)	Input Variable – It represents the total surface area of the roof of the building.
Overall Height	quantitative	meters (m)	Input Variable-- It represents the vertical distance from the base to the highest point of the building.
Orientation	quantitative	Number	Input Variable-- It refers to the direction in which the building is facing or positioned, such as north, south, east, or west.

Glazing Area	quantitative	square meters (m^2)	Input Variable— It represents the total area of windows and other transparent or translucent surfaces in the building envelope.
Glazing Area Distribution	quantitative	percentage or a fraction	Input Variable – It is a measure of how the glazing area is distributed or located on different facades of the building.
Heating Load (HL)	quantitative	kilowatts (kW)	Output Variable - The heating load is a measure of the amount of heat energy required to maintain a comfortable indoor temperature during the heating season
Cooling Load (CL)	Quantitative	Kilowatts (kW)	Output Variable - The cooling load is a measure of the amount of cooling capacity required to maintain a comfortable indoor temperature during the cooling season

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.

d) Perform Principal Component analysis and convert the features into principal components.

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

d) Perform Principal Component analysis and convert the features into principal components.

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.

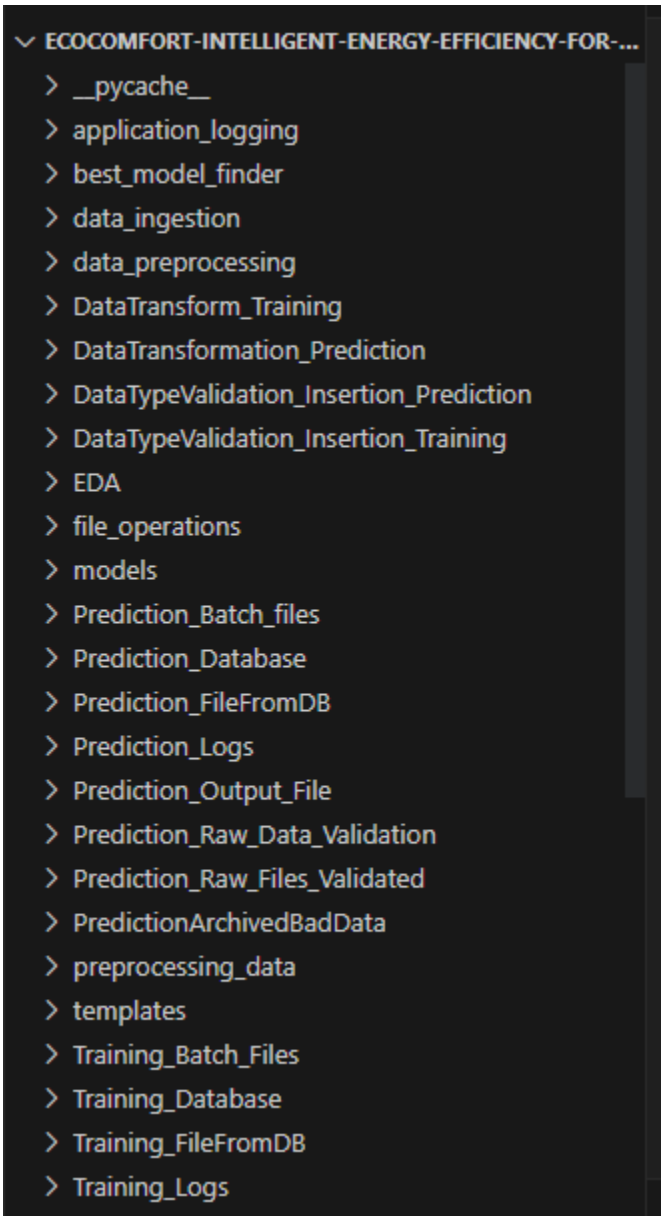
5) Once the prediction is made for all the clusters, the predictions along with the original names before label encoder are saved in a CSV file at a given location and the location is returned to the client.

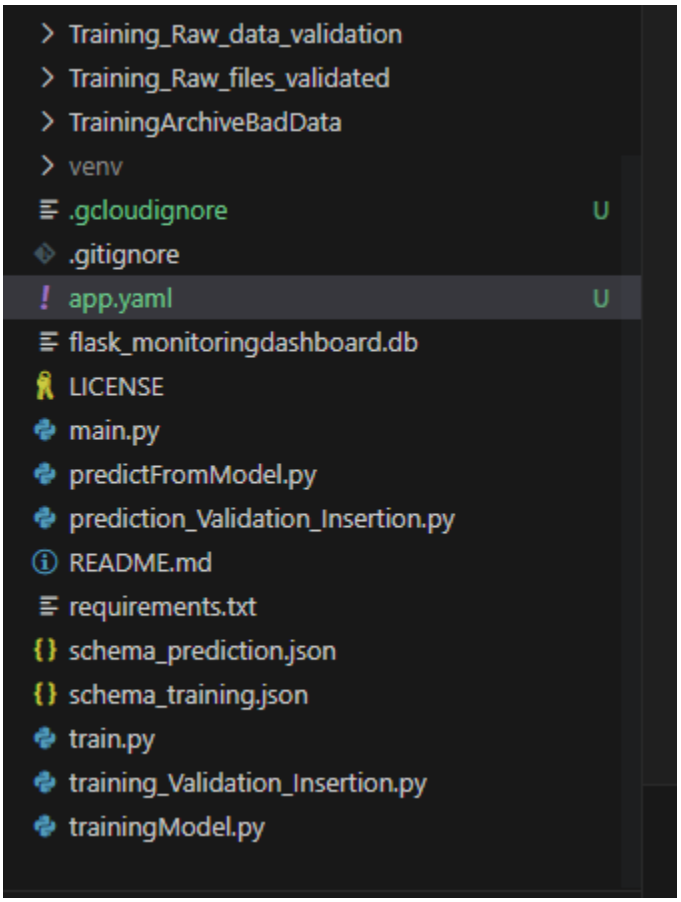
Deployment

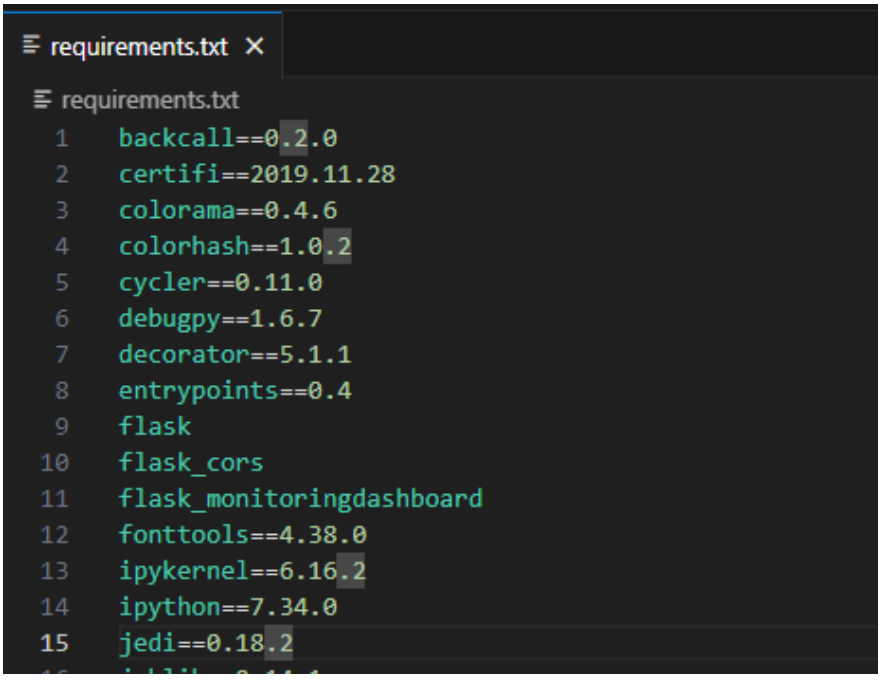
We will be deploying the model to the Google Cloud Platform.

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

Now let's see the "EcoComfort: Intelligent Energy Efficiency" project folder structure.


ECOCOMFORT-INTELLIGENT-ENERGY-EFFICIENCY-FOR-...
__pycache__
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





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.

predictFromModel.py

```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.

app.yaml

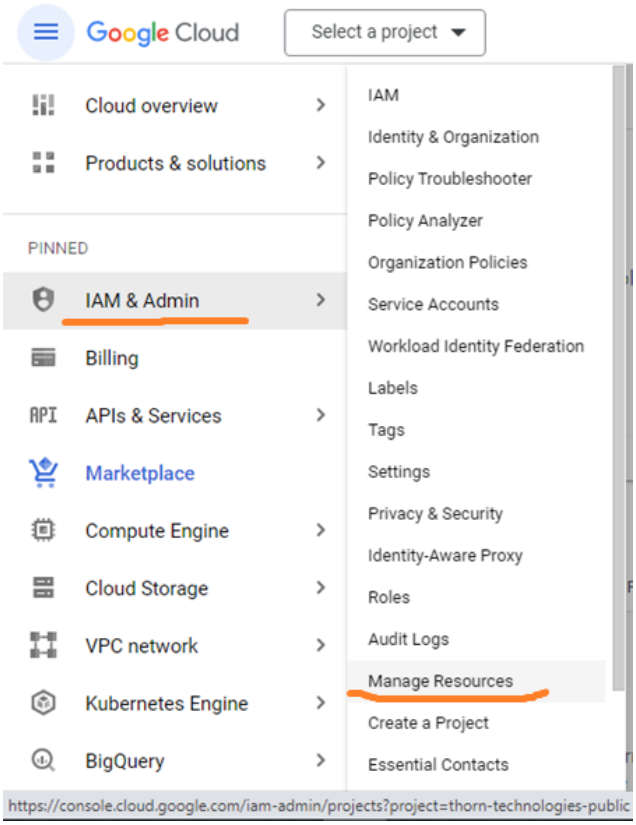
```
runtime: python37
```

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

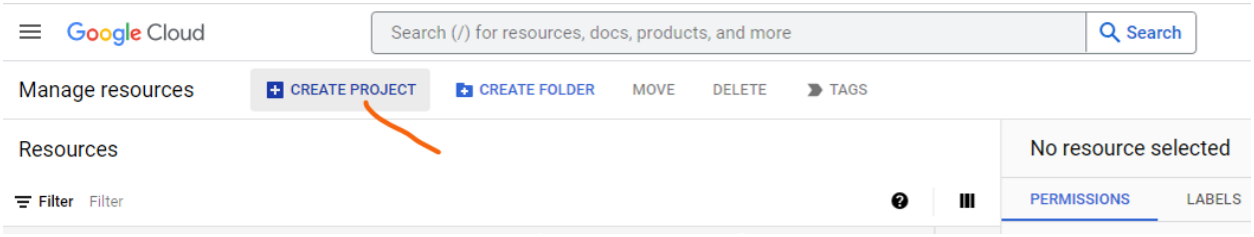
Deployment to Google cloud (GCP)

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/

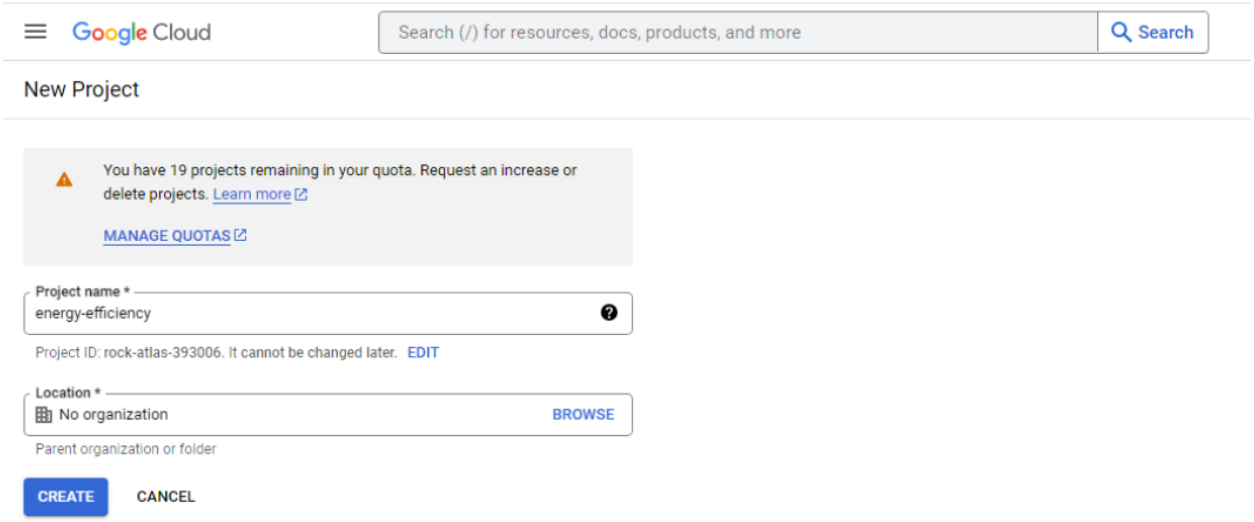
2. Go to IAM & Admin and select Manage Resources



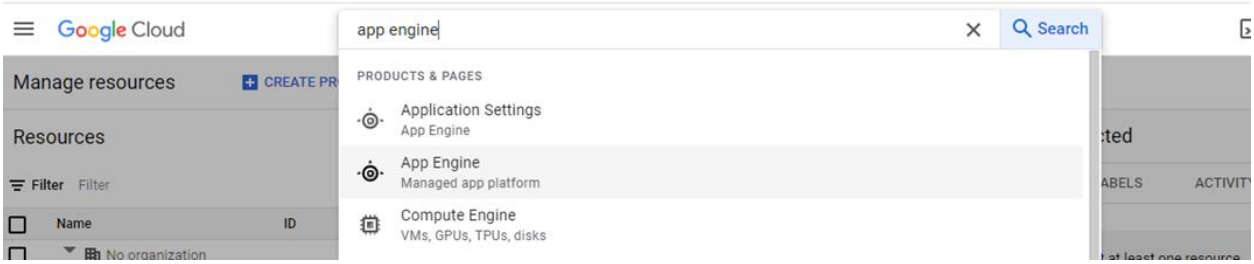
3. In the manage resources page, click on CREATE PROJECT option.

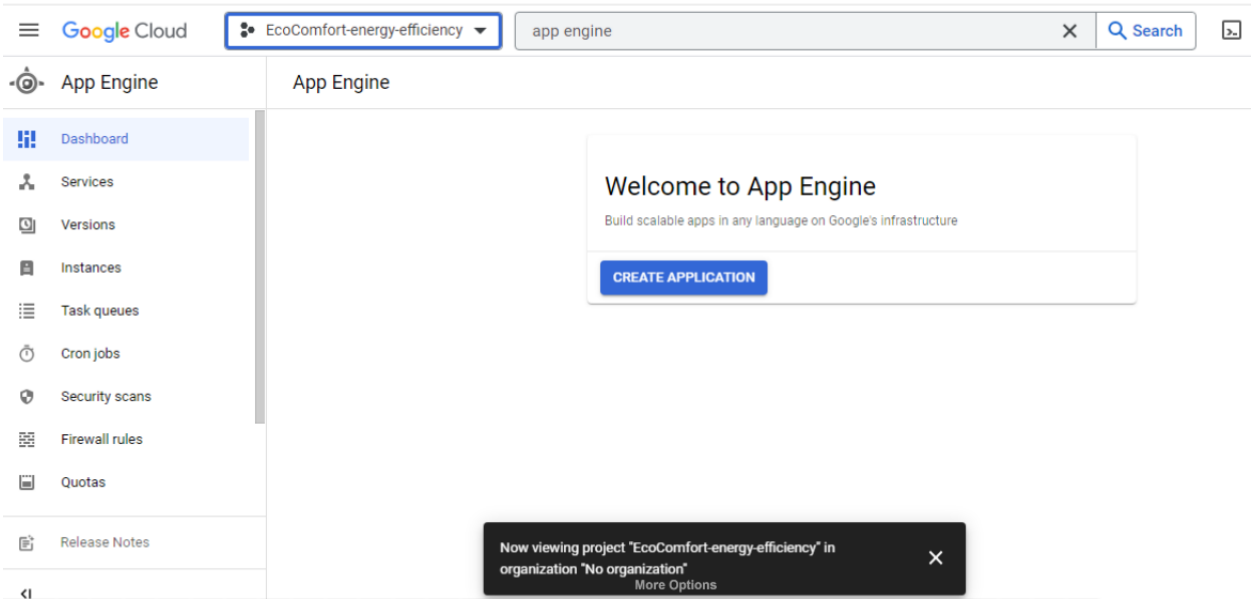


4. Give your project name and Location details. And click on CREATE.



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

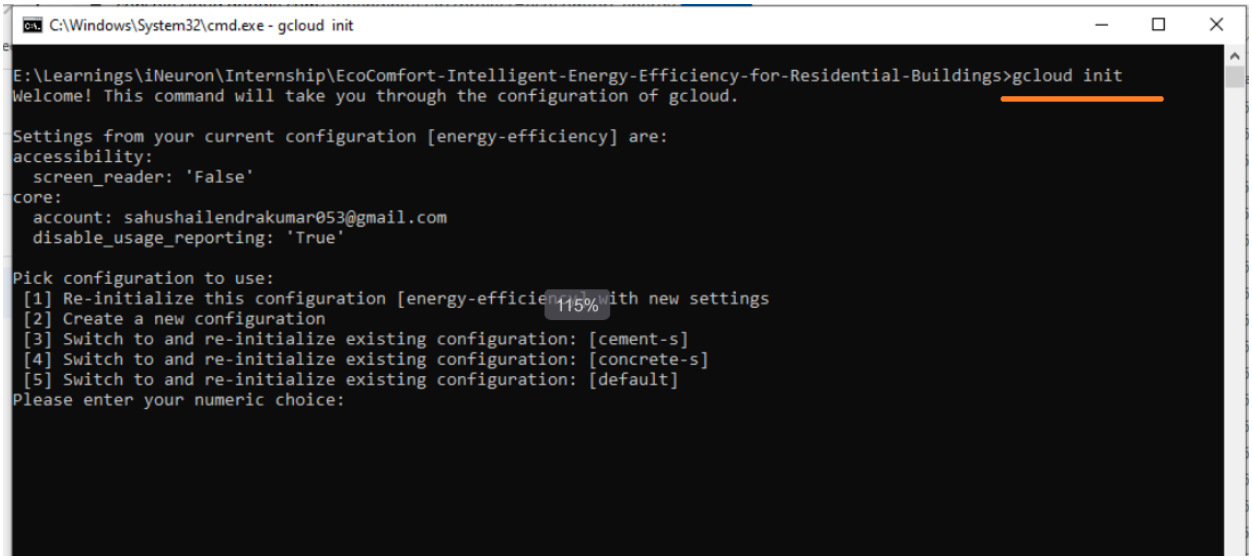




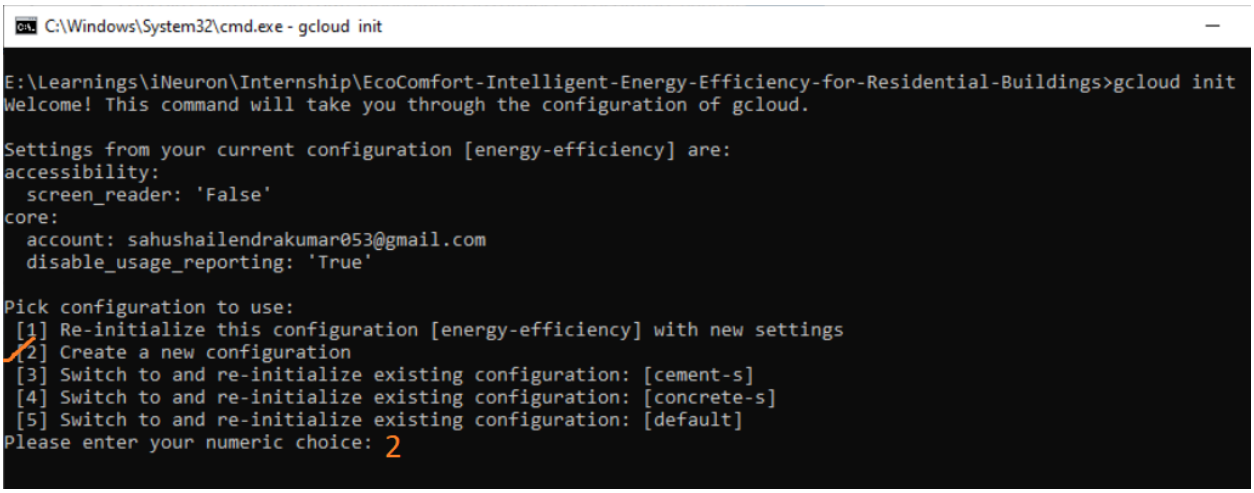
6. Go to https://dl.google.com/dl/cloudsdk/channels/rapid/GoogleCloudSDKInstaller.exe to download the google cloud SDK in your machine. And install it.

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.

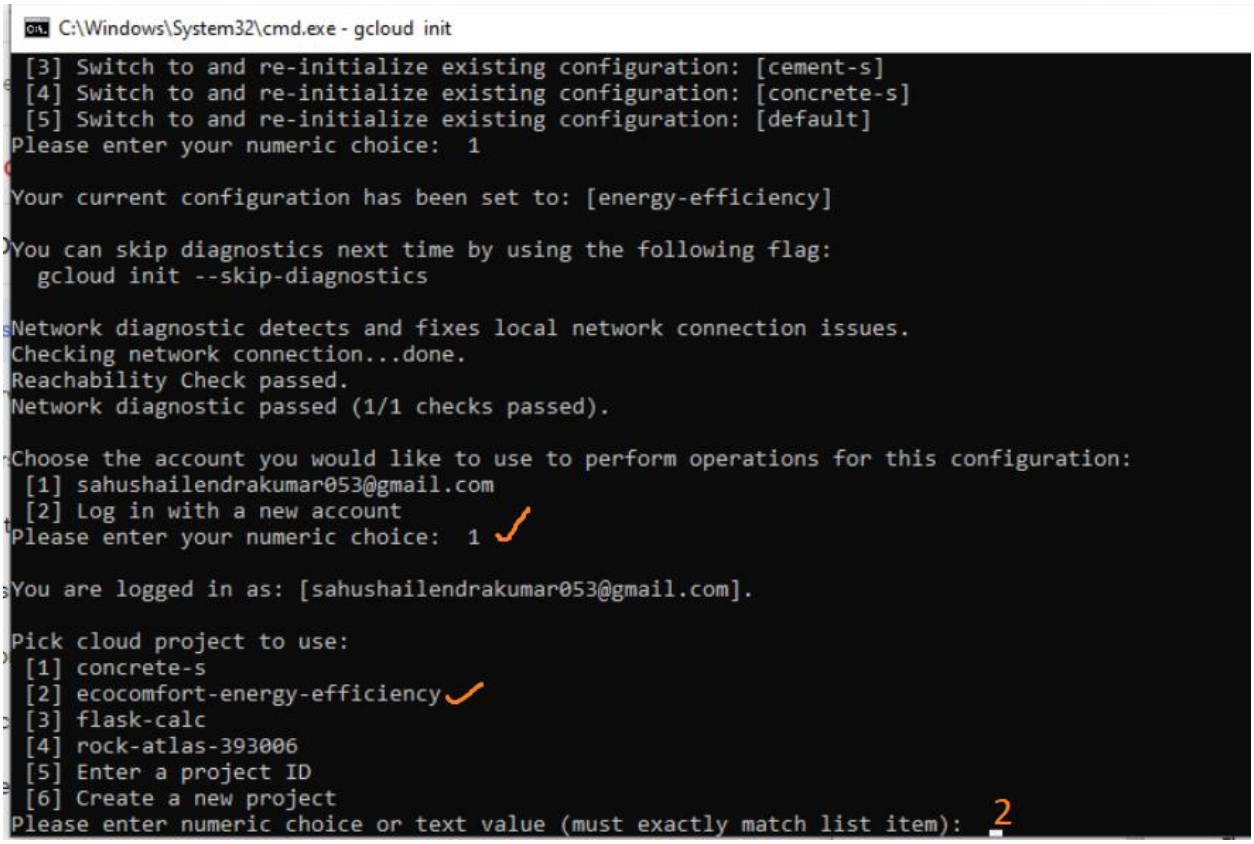


9. Select option 2 for new configuration and then write the name of the project that you created previously.



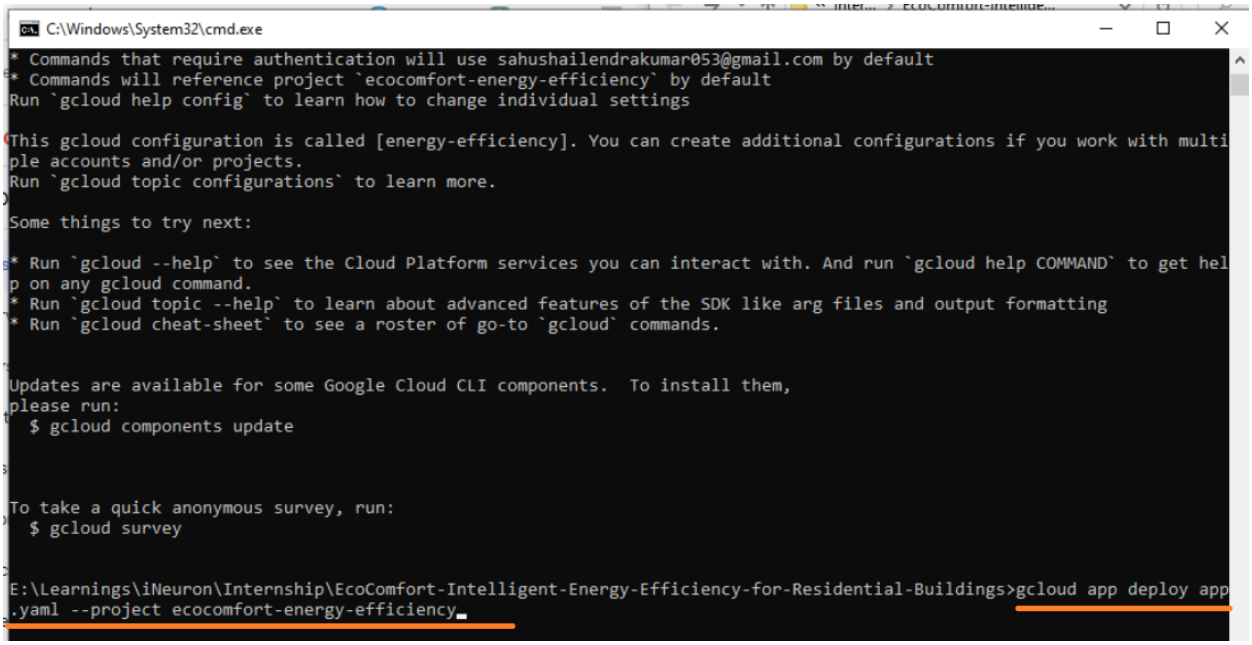
10. Select the account that you want to use or login with another.

    Then pick the project name to use.

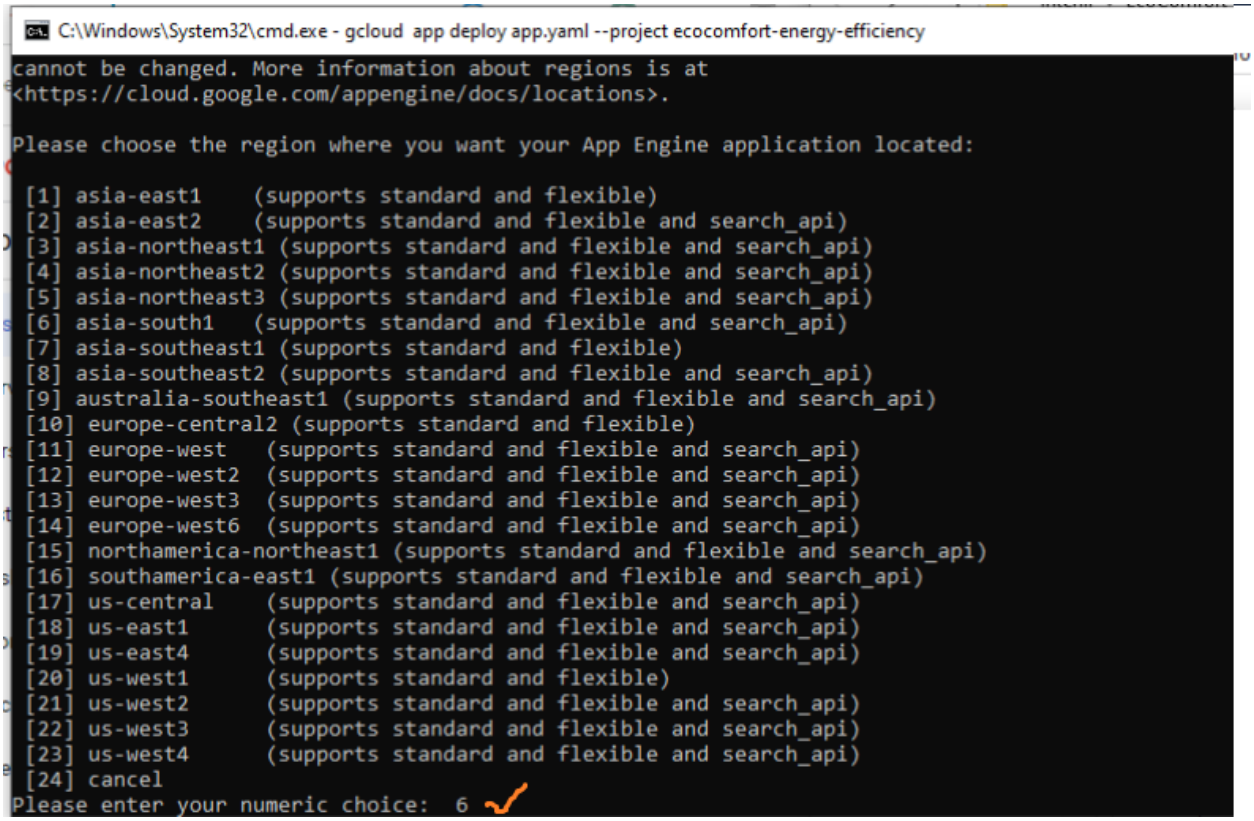


11. Run this command to deploy the project

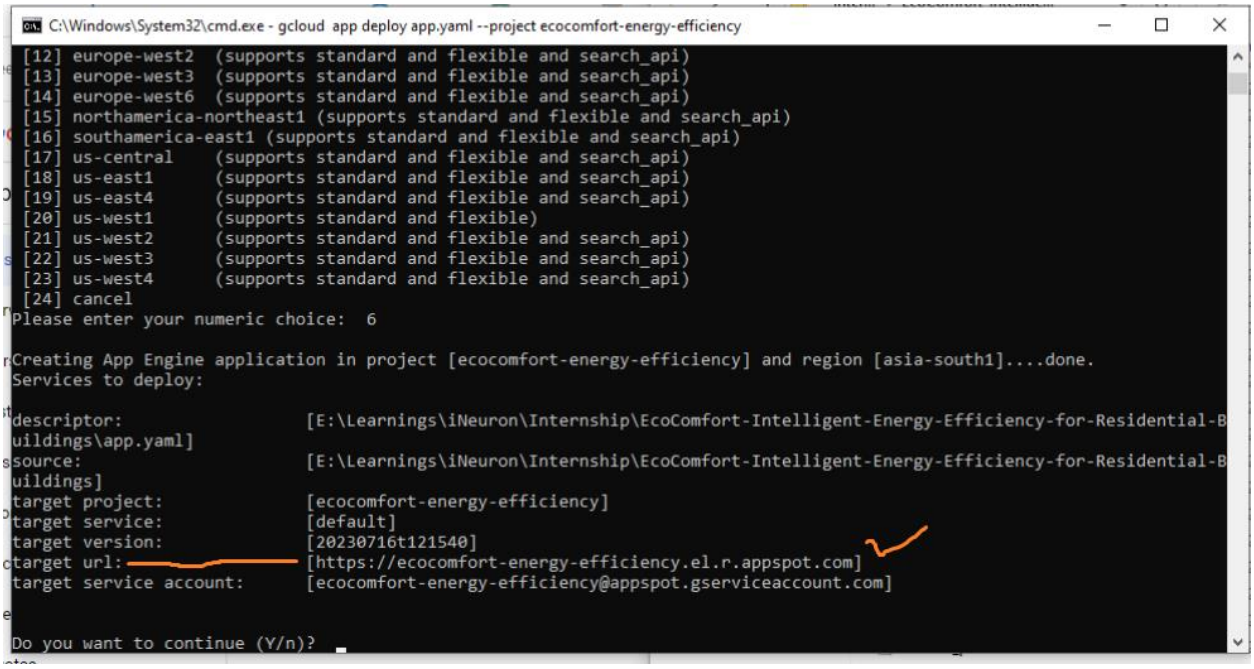
    gcloud app deploy app.yaml --project <<project name>>



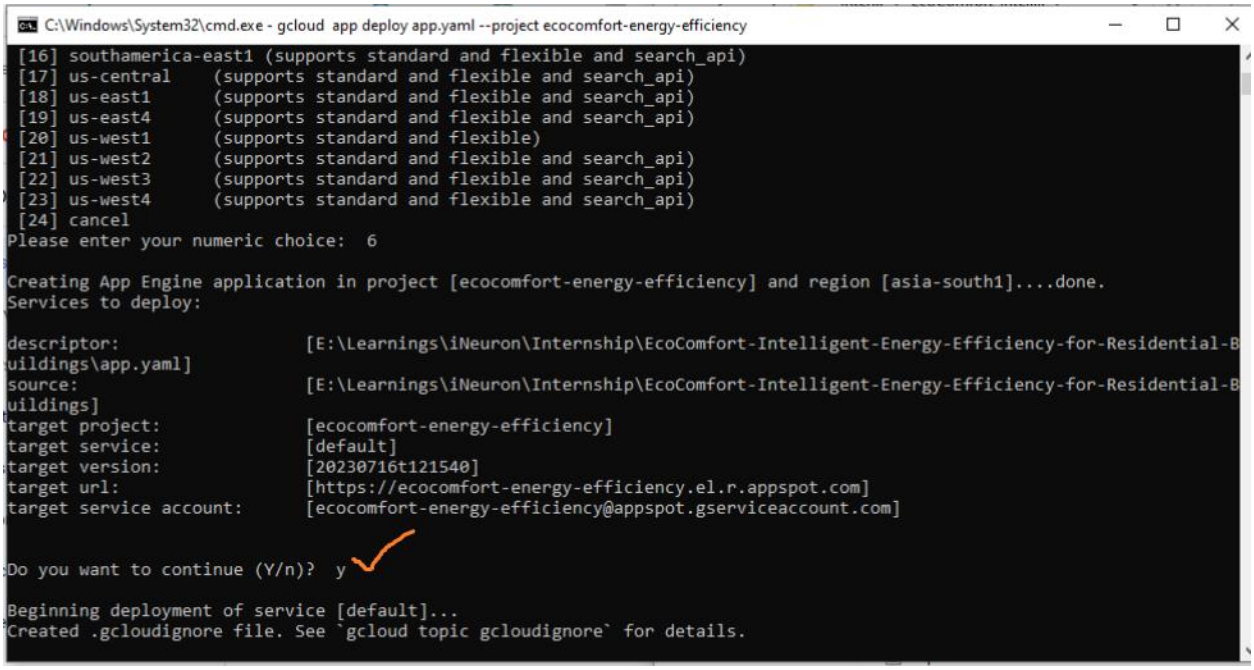
12. Now choose your region.



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

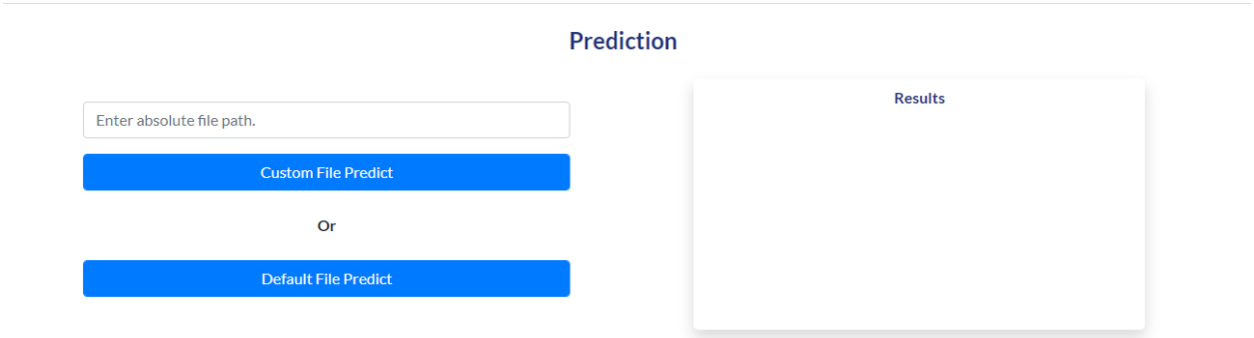


14. It will ask to continue uploading all the files into cloud.
    Select Y to give it yes to continue.



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 the given link to open it in the browser.

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.