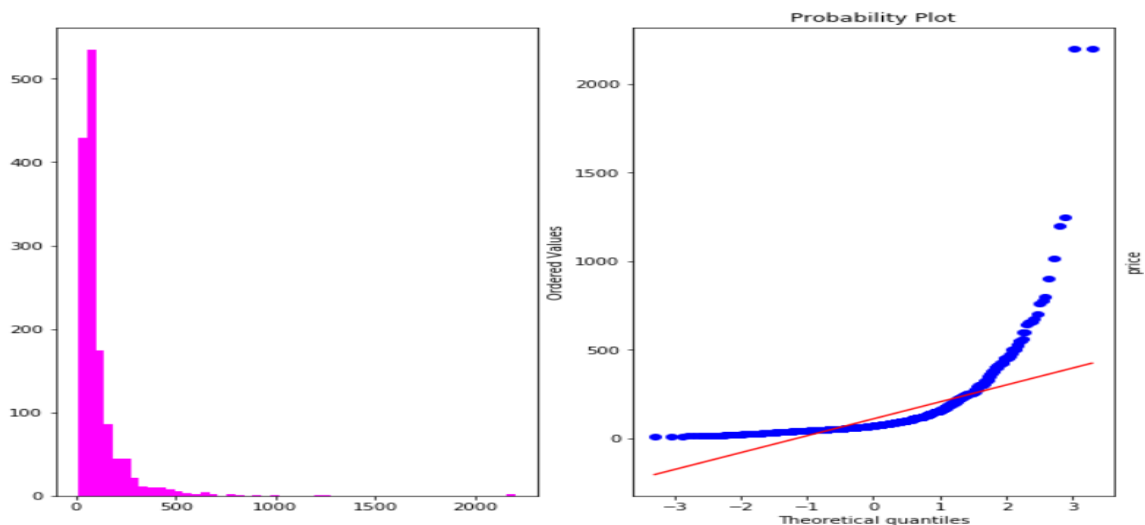


Hyderabad House Price Prediction

Here in this project our team has focused on predicting the price of the Houses present in different locations, here the parameters/columns which we have used are area_type, location, number of rooms, total square feet area, total number of bathrooms, total number of balconies, based on these parameters the Machine Learning Model is going to predict the Price of the Particular House, here is a quick glimpse of the dataset

0	area_type	1402	non-null	object
1	location	1402	non-null	object
2	size	1401	non-null	object
3	total_sqft	1402	non-null	object
4	bath	1391	non-null	float64
5	balcony	1331	non-null	float64
6	price	1402	non-null	float64

As we can observe from the dataset, the area_type is of Object type, which is a Categorical Feature, location, size, total square feet, price are all Continuous variables, and balcony is an Ordinal Variable, even though its datatype is mentioned as float, it is not necessary that variables having int/float datatype be always continuous, they may be ordinal as well as we can observe in this case, any how as we are using the AUTOAI feature of IBM Watson, this is automatically handled by them, on other hand let us try to visualise some statistical trends of the continuous variables, as shown ↓

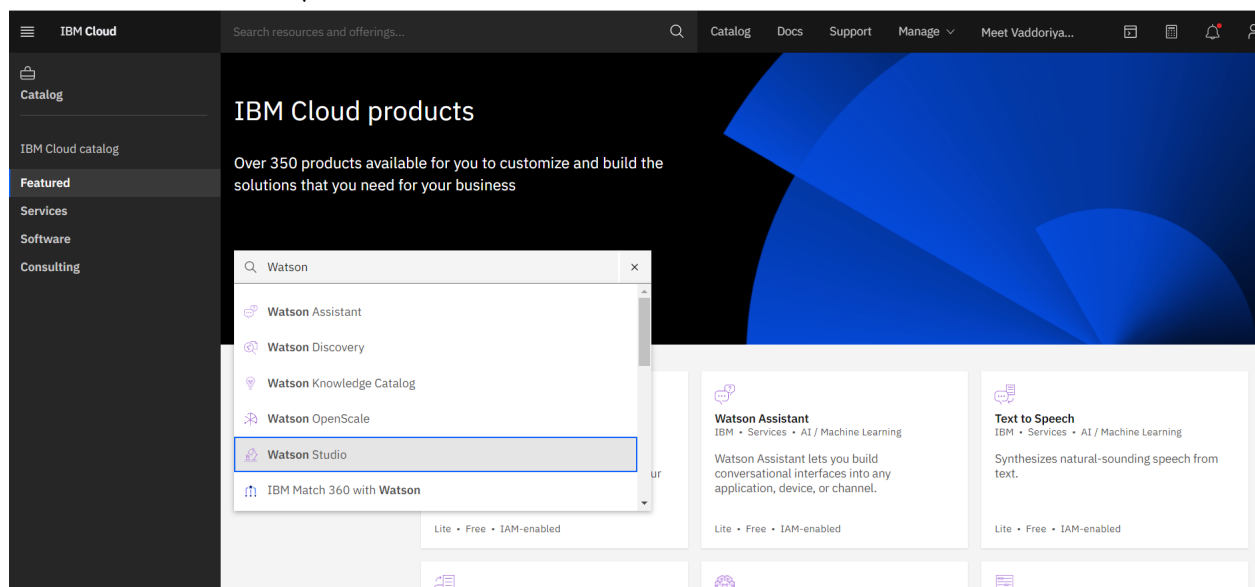


So these are the histogram plot and the Quantile-Quantile Plots relating to the Price variable which is continuous, now from the above Distribution we can observe that the distribution is skewed towards the left, and also the Quantile-Quantile plot indicates that the line is not perfectly fitted on the dataset, if the line is fitting that dataset, then we can say that the data is normalised, or the data follows "Gaussian Distribution", here a lot of statistical transformations are required to make such a skewed data into a Gaussian Distributed, like Boxcox Transformation, Chi Square transformation etc, one can refer these links which will clearly explain about the [Q-Q plots](#) and the Gaussian Distribution [Gaussian Distribution](#), as we are working on "AUTO AI" model of IBM Watson, this will automatically set the best distribution for all the features.

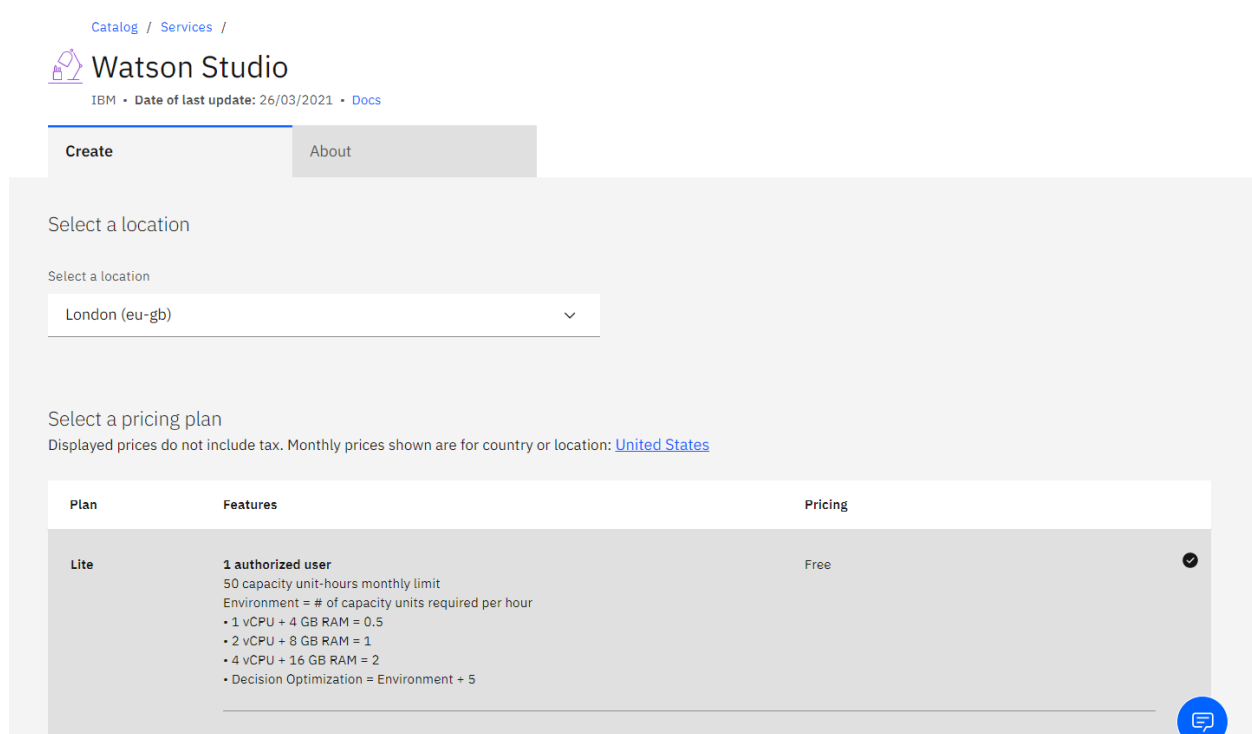
Another important step involved is about handling the categorical features, as the Machine Learning Model only knows about how to learn numerical data, it has no idea about a category, so we have to encode the categorical variables into numeric form, as we have done on IBM Watsons' AUTO AI, this will automatically handle the terms

Now let's discuss about how to start with AUTO AI

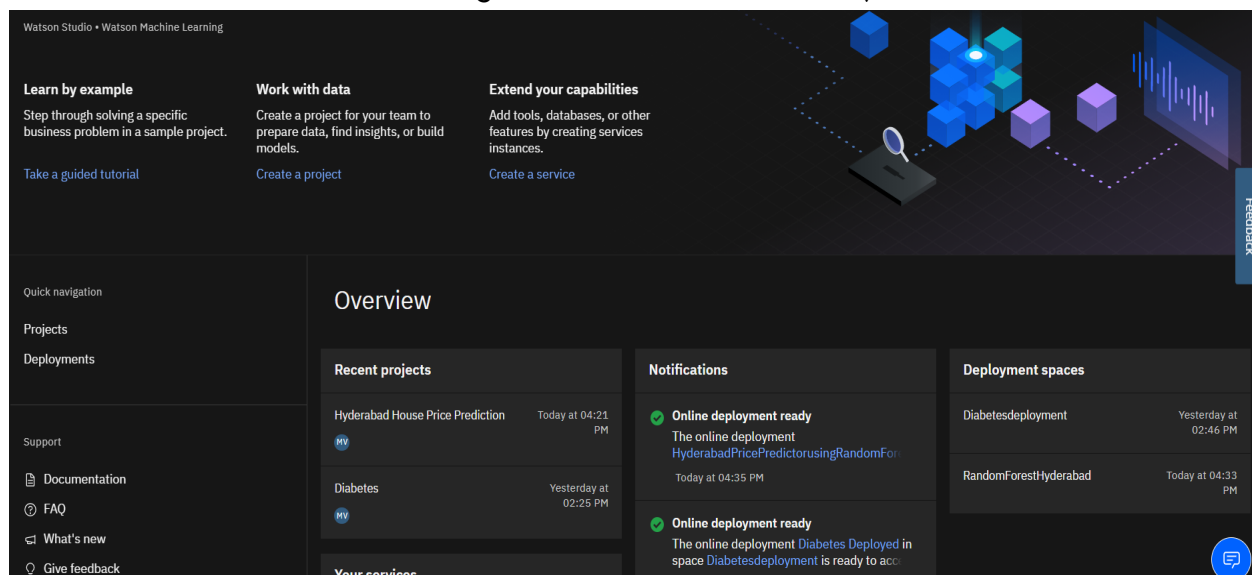
First we need to create an IBM Watson Instance which we can create by logging in on [IBM Cloud](#), then after logging in, we have to create a resource, the icon will be present on the top-left, of the dashboard, so after selecting "Create Resource" we will be getting an interface as follows ↓



Then we have to type in "Watson Studio" and then select the Watson Studio, after that the IBM Watsons' Interface will be visible, it will look something like ↓

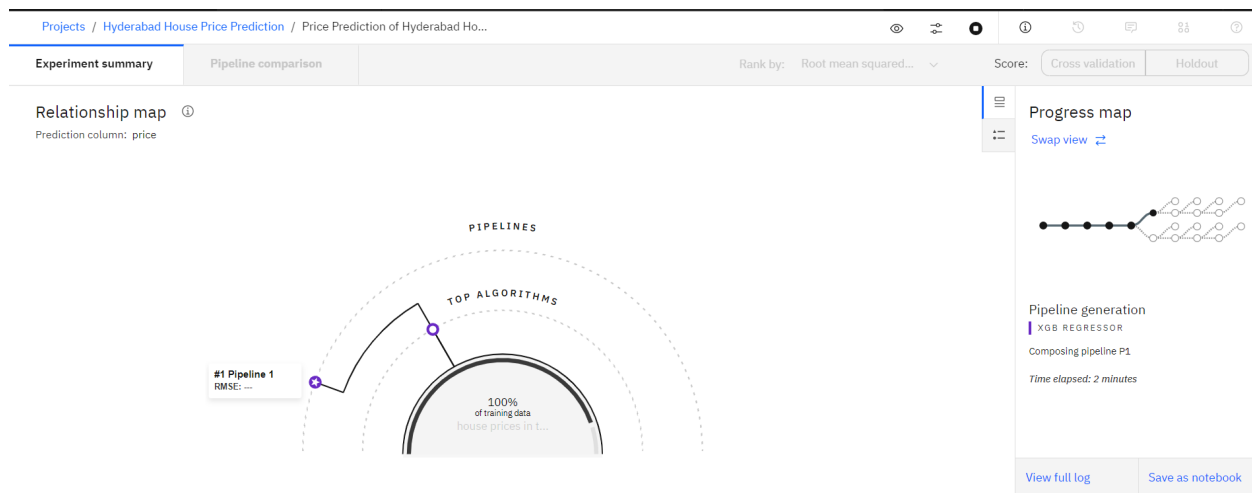


There we will be having a "CREATE" button, we have to simply click on that and our Watson Studio will be created successfully! Then, once we're done with that, we have to switch ourselves from "ibmcloud" to "[IBM WATSON STUDIO](#)", where we have to enter our details and then we will be looking at an interface as shown ↓



then once done, we have to know start making our project, for this you can follow the steps discussed by [IBM Developer](#)

Now here is the Model Predictions Made by AUTO AI in case of predicting the "Price"



Pipeline leaderboard

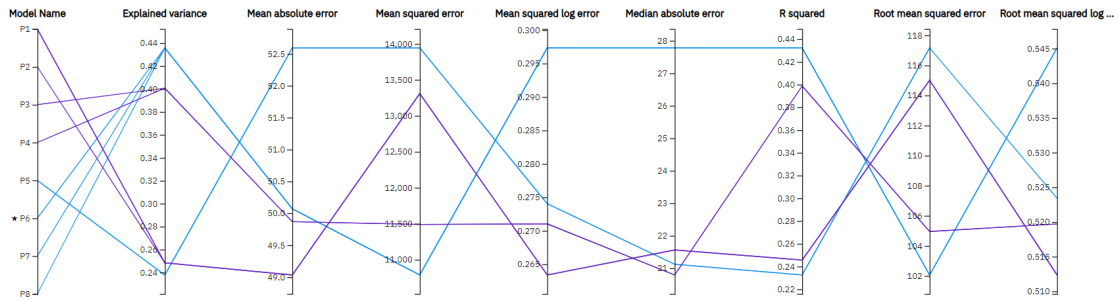
Pipeline leaderboard

Rank	↑	Name	Algorithm	RMSE (Optimized)	Enhancements	Build time
★ 1		Pipeline 6	Random Forest Regressor	102.060	HPO-1	00:00:05 Save as
2		Pipeline 7	Random Forest Regressor	102.060	HPO-1 FE	00:00:32
3		Pipeline 3	XGB Regressor	104.951	HPO-1 FE	00:00:20
4		Pipeline 4	XGB Regressor	104.951	HPO-1 FE HPO-2	00:00:42
5		Pipeline 1	XGB Regressor	115.010	None	00:00:01
6		Pipeline 2	XGB Regressor	115.010	HPO-1	00:00:10
7		Pipeline 5	Random Forest Regressor	117.161	None	00:00:01

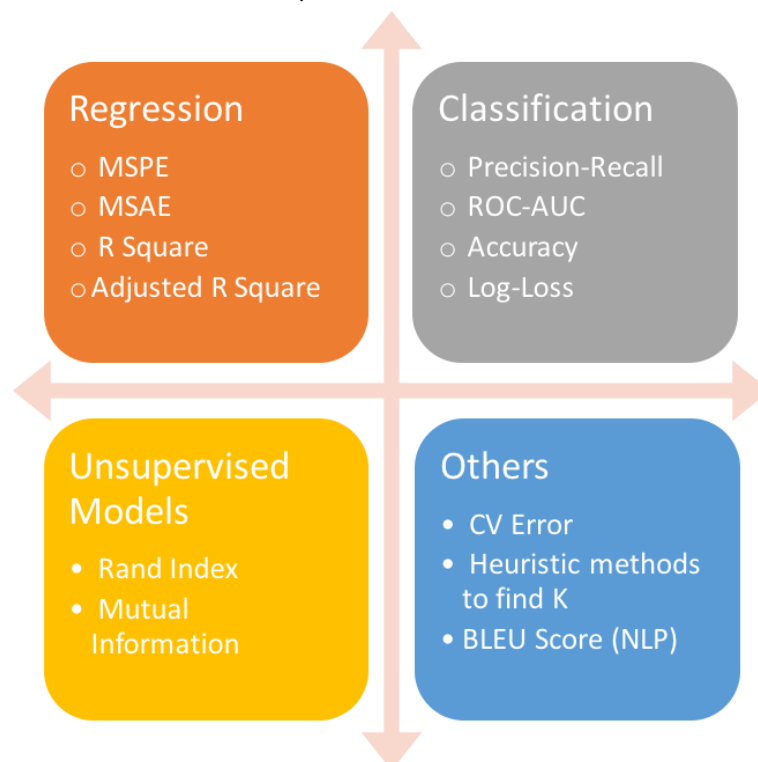
Here we can see that AUTO AI has created almost 6 pipelines for predicting the price and the one which gave the best result has been chosen, we can see that after hyperparameter tuning, K Fold Cross Validation and other stuff, Random Forest Regressor gave us the best result, well the metric which we had chosen was "Root Mean Squared Error", so based on this metric Random Forest Regressor gave us the best results we can also check out other metrics as well, by clicking on the metric chart,

Metric chart ①

Prediction column: price



we can analyse various metrics as per our need here are few commonly used metrics in case of Regression and Classification,



After analysing the model and its metrics, now it's time for deploying the model, so for that we have to follow the given steps ↓

Deployments / RandomForestHyderabad / Price Prediction of Hyderabad Ho...

Price Prediction of Hyderabad Houses - RandomForestRegressorEstimator

Create deployment

Deployments Schema

DEPLOYMENT TYPES		1 Online Deployment(s)		
		Name	Status	Last modified
Online	(1)			
Batch	(0)	HyderabadPrice...	In progress	Apr 9, 2021 4:34 PM

Price Prediction of Hyderabad Houses - RandomForestRegressorEstimator

Last modified at Apr 9, 2021 4:34 PM

Created
Apr 9, 2021 4:32 PM

Type
wml-hybrid_0.1

Model ID
738b593d-c2ec-455a-a6f1-e4d...

Software specification
hybrid_0.1

Hybrid pipeline software specifications
autoai-kb_3.1-py3.7

Description
No description provided.

Tags

After clicking on "Create Deployment", the steps are easy you have to just click on create create etc..and once all of this is done, we have to select the "ONLINE" mode for deployment, as we are gonna use the "Flask" which is a Webframework supported in python, so after selecting the "Online" Mode, we will be getting an interface as shown ↓

Deployments / RandomForestHyderabad / Price Prediction of Hyderabad Ho... / HyderabadPricePredictorusingRandomForest

HyderabadPricePredictorusingRandomForest Deployed Online

API reference Test

Direct link

Endpoint

Bearer <token>

https://eu-gb.ml.cloud.ibm.com/ml/v4/deployments/e96f36c2-d525-4f61-8b19-4ebf3b1bb5ab/predictions?version=2021-04-01

Code snippets

cURL Java JavaScript Python Scala

```
import requests

# NOTE: you must manually set API_KEY below using information retrieved from your IBM Cloud account.
API_KEY = <your API key>
token_response = requests.post('https://iam.cloud.ibm.com/identity/token', data={"apikey": API_KEY, "grant_type": "urn:ibm:params:oauth:grant-type:apikey"}, headers={"Content-Type": "application/json"})
mltoken = token_response.json()["access_token"]

header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + mltoken}

# NOTE: manually define and pass the array(s) of values to be scored in the next line
payload_scoring = {"input_data": [{"fields": [array_of_input_fields], "values": [array_of_values_to_be_scored, another_array_of_values_to_be_scored]}]}

response_scoring = requests.post('https://eu-gb.ml.cloud.ibm.com/ml/v4/deployments/e96f36c2-d525-4f61-8b19-4ebf3b1bb5ab/predictions?version=2021-04-01', headers=header, json=payload_scoring)
```

HyderabadPricePredictorusingRandomForest

Created
Apr 9, 2021 4:34 PM

Updated
Apr 9, 2021 4:34 PM

Deployment ID
e96f36c2-d525-4f61-8b19-4eb...

Software specification
hybrid_0.1

Hybrid pipeline software specifications
autoai-kb_3.1-py3.7

Copies
1

Description
No description provided.

Tags
Add tags to make assets easier to find.

Associated asset
Price Prediction of Hyderabad Ho...

as we can observe there is some code given by default by AUTOAI, so we have to copy that code and in the "app.py" file which is the python file we have to paste this following code, before that in order to access the services of the IBM services we need to have an "API" key, so for this we have to login back to IBM cloud and from there we have to find our API key, every user has a unique API key for every project which is using cloud services, so the interface of API keys will look something like this ↓

API keys

Create, view, and work with API keys that you have access to manage. IBM Cloud API keys are associated with a user's identity and can be used to access cloud platform and classic infrastructure APIs, depending on the access that is assigned to the user. The following table displays a list of API keys created in this account. [Learn more.](#)

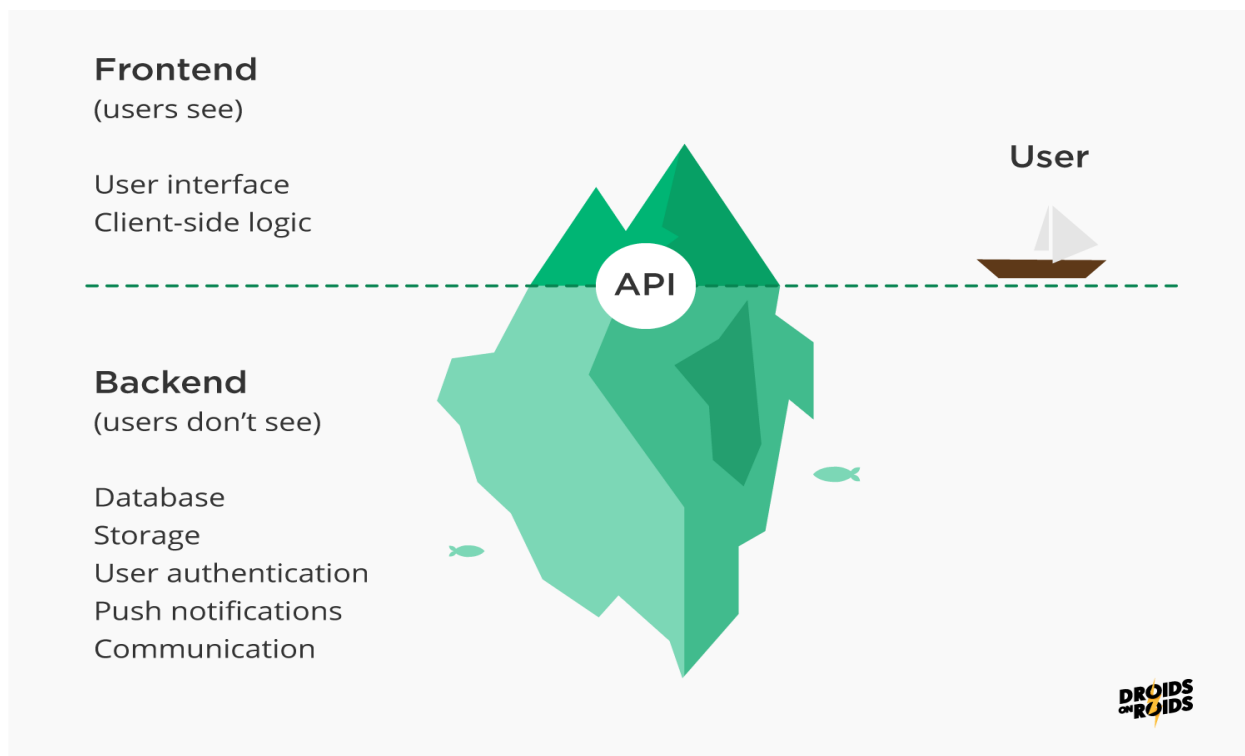
Looking for more options to manage API Keys? Try [IBM Cloud® Secrets Manager](#) for creating and leasing API keys dynamically and storing them securely in your own dedicated instance.

View: My IBM Cloud API keys

API keys associated with a user's identity have the same access that the user is assigned across all accounts. To update the access for an API key, assign or remove access for the user.

					Create an IBM Cloud API key +
Status	Name ↓	Description	Date Created		
🔒	Diabetes key		2021-04-08 10:18 GMT		⋮
🔒	HyderabadPricePrediction		2021-04-09 11:09 GMT		⋮
Items per page: 25		1-25 items	Page 1		

Then we have to click on the "Create IBM Cloud API key" and we will be getting the API key, which we have to use it in the flask app, here is an image which will make it clear



So here the User will only be able to see the "User Interface", which is nothing but "Front-end" of the project, and what ever the user interacts with the UI, like filling in a form and etc, all those are sent as a request to the "Backend!", now what's this API, here it's like a connection of the frontend and the backend, here in this case in backend I have used "Flask" which is a web framework based on python, and it's through this backend the UI will interact with the user, so as we have used "IBM Watson Studio", and our project is hosted or deployed on "IBM cloud", so this is kind of backend, now whenever the user fills in a form, hits enter, the data of the form is first reached at the flask server, which is a POST request, then once the server fetches the data, it passes that data to the cloud, where in our Machine Learning Model is hosted, this model makes a prediction and returns the result back to the Flask server which in turn will return the result to the UI, here is the step by step procedure for that.

```
def predict():
    if request.method == 'POST':
        my_dict = request.form

        area_type = str(my_dict['areatype'])
        location = float(my_dict['location'])
        roomsize = float(my_dict['roomsize'])
        sqftarea = float(my_dict['sqftarea'])
        washrooms = float(my_dict['washrooms'])
        balcony = float(my_dict['balcony'])

        input_features = [[area_type, location, roomsize,
                             sqftarea, washrooms, balcony]]

        # input = ["area_type", "location", "size", "total_sqft", "bath", "balcony"]

        payload_scoring = {"input_data": [{"fields": [
            "area_type", "location", "size", "total_sqft", "bath", "balcony"]}, {"values":
            input_features}]}

        response_scoring = requests.post('https://eu-gb.ml.cloud.ibm.com/ml/v4/deployments/
e96f36c2-d525-4f61-8b19-4ebf3b1bb5ab/predictions?version=2021-04-09',
                                         json=payload_scoring, headers={'Authorization': 'Bearer ' +
```

so as I said after the user has interacted with the UI i.e. by filling the form, we send the details filled by the user to the backend, we collect all the data from the form and store it in a list, which is "input_features", now the main part of the whole project!, we pass in this list to the cloud so that we can get the predicted price, so as we had copied the code which "AUTO AI" had given us, it's time to use it now, so we pass in that list to that payload scoring and this ensures that it gets all the parameters of the form, now the next step is to send the data to the model which is deployed on IBM Cloud, so we send a POST request to the API, which we had taken from the API keys, so once we pass the data through API to the Machine Learning Model which we have created on the IBM cloud, then we try to fetch out the results which the Machine Learning Model had sent


```

response_scoring = requests.post('https://eu-gb.ml.cloud.ibm.com/ml/v4/deployments/
e96f36c2-d525-4f61-8b19-4ebf3b1bb5ab/predictions?version=2021-04-09',
                                json=payload_scoring, headers={'Authorization': 'Bearer ' +
mltoken})

print("Scoring response")
predictions = response_scoring.json()
print(predictions)
# return render_template('predict.html')

# {'predictions': [{'fields': ['prediction'], 'values': [[73.1356229374655]]}]}
final_price = predictions['predictions'][0]['values'][0][0]

final_price = round(final_price, 2)
# print(f' The value is : {final_price}')

```

we get the response from the Cloud, we do some small modifications, as the data which is returned by cloud is in form of "json" format which we have to convert in python list, and then access the items from the list, after fetching the result we have to return it to the UI, and that will show the result as ↓

House Price Predictor

Based on the details filled the predicted price for your house is : 138.84 Lakh

[Go Back](#)