gRPC Basics – Python

This tutorial provides a basic Python programmer’s introduction to working with gRPC.

By walking through this example you’ll learn how to:

* Define a service in a .proto file.
* Generate server and client code using the protocol buffer compiler.
* Use the Python gRPC API to write a simple client and server for your service.

It assumes that you have read the [Overview](https://grpc.io/docs/guides/#overview) and are familiar with [protocol buffers](https://developers.google.com/protocol-buffers/docs/overview). You can find out more in the proto3 language guide and Python generated code guide.

Why use gRPC?

This example is a Machine Learning Regression example that lets clients get the sales prediction based on choosen attributes.

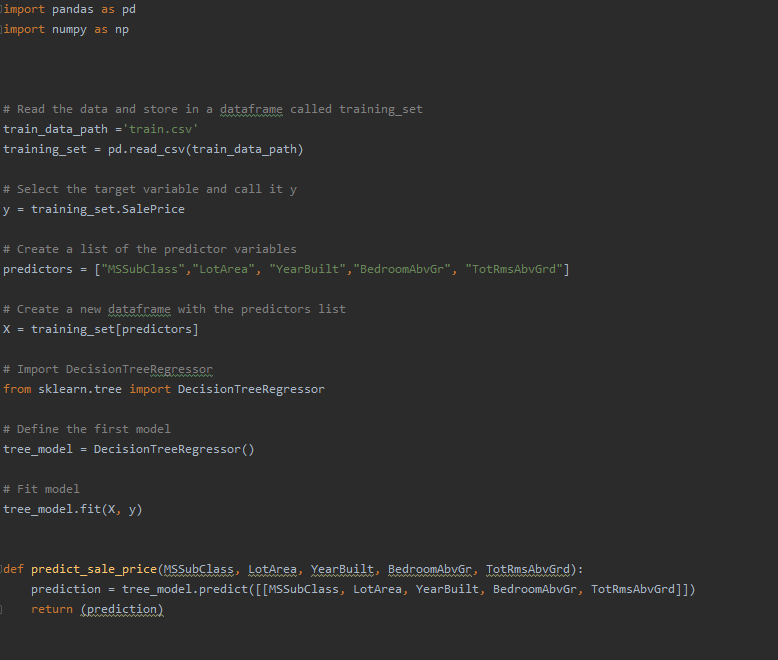
With gRPC you can define your service once in a .proto file and implement clients and servers in any of gRPC’s supported languages, which in turn can be run in environments ranging from servers inside Google to your own tablet - all the complexity of communication between different languages and environments is handled for you by gRPC. You also get all the advantages of working with protocol buffers, including efficient serialization, a simple IDL, and easy interface updating.

Steps

1. Write the service to be served.
2. Make a proto file to define the messages and services.
3. Use the proto file to generate gRPC classes for Python
4. Create the server.
5. Create the client.
6. Prepare the docker file and run the docker.

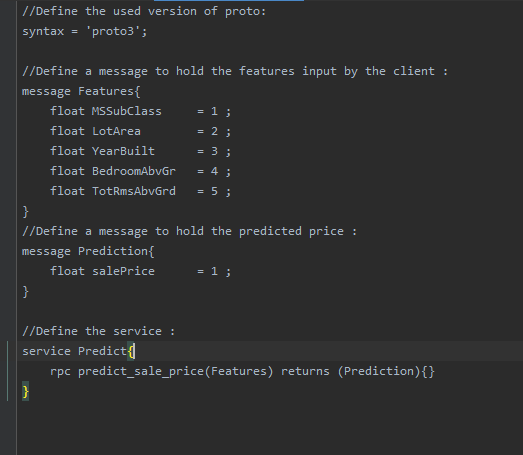
Step 1: Write the Service:

In our case, the service is predicting house pricing. Below is code snippet.



This model has 5 input arguments (features of the house that we want to predict its sale price)

Step 2: Make the Proto File:



Here, we did not give values to the features, those numbers indicate the order of serializing the features.

## Step 3: Generate gRPC classes for Python:

Open the terminal, change the directory to be in the same folder that the proto file is in.

To generate the gRPC classes we have to install the needed libraries first:

**#Install gRPC :**

python3 -m pip install grpcio

**#To install gRPC tools, run:**

Python3 -m pip install grpcio-tools googleapis-common-protos

**Now, run the following command:**

Python3 -m grpc\_tools.protoc -I. --python\_out=. --grpc\_python\_out=. ML\_example.proto

This command used ML\_example.proto file to generate the needed stubs to create the client/server.

The files generated will be as follows:

**ML\_example\_pb2.py — contains message classes**

* **ML\_example\_pb2**.**Features** for the input features
* **ML\_example\_pb2**.**Prediction** for the prediction price

**ML\_example\_pb2\_grpc.py — contains server and client classes**

* **ML\_example\_pb2\_grpc**.**PredictServicer** will be used by the server
* **ML\_example\_pb2\_grpc**.**PredictStub** the client will use it

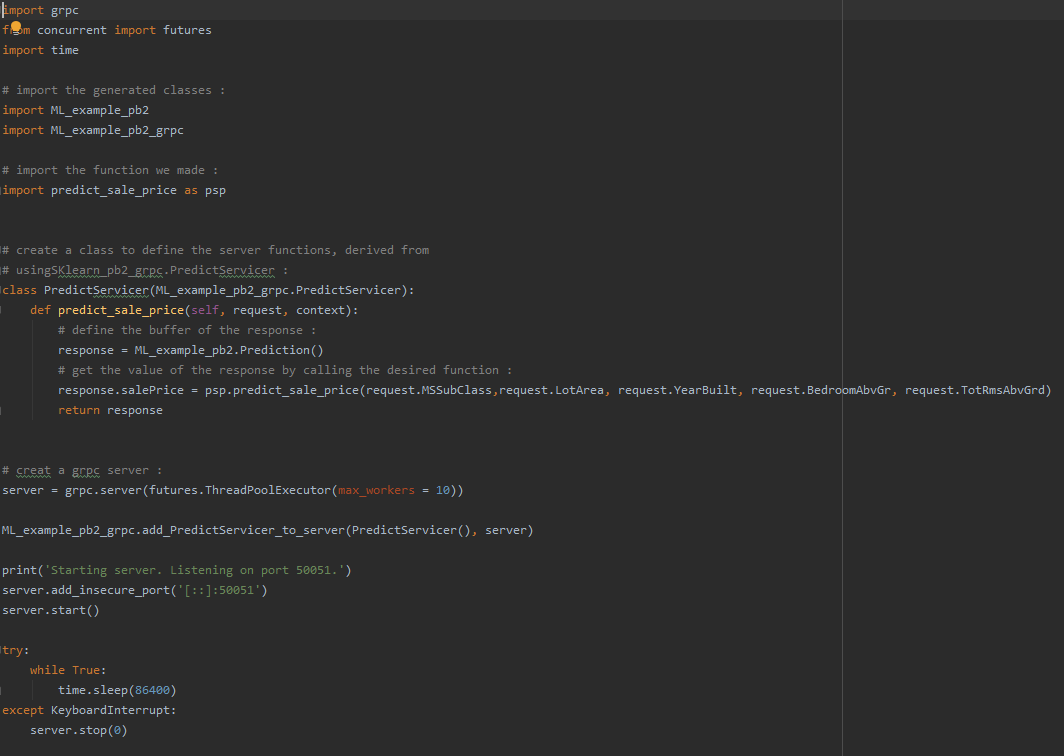
## Step 4: Creating the Server:

The server will import the generated files and the function that will handle the predictions.

Then we will define a class that will take a request from the client and uses the prediction function to return a respond.

The request gives us the five features, the respond is a prediction.

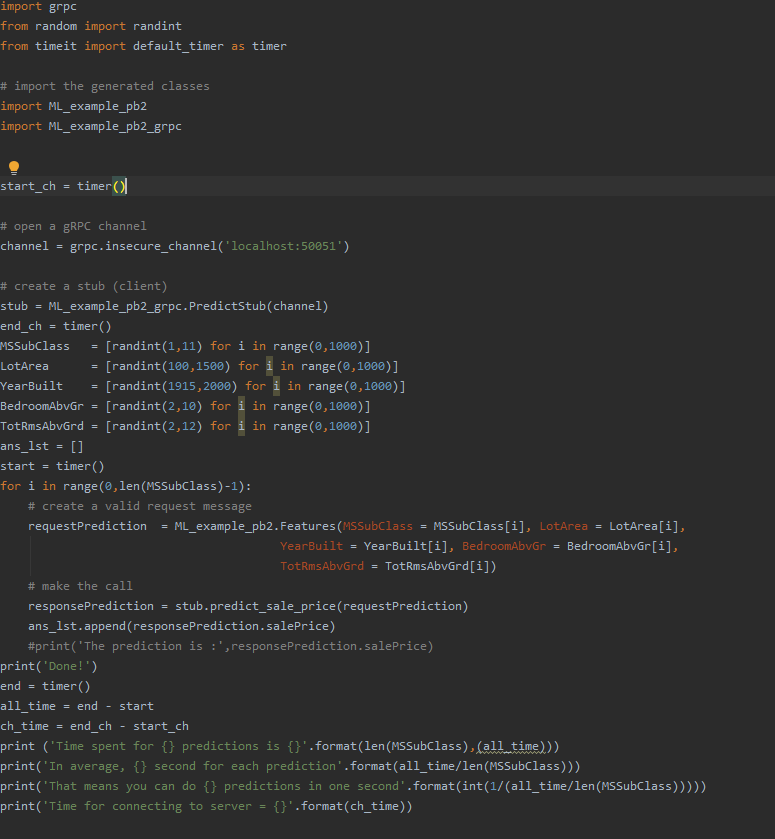
After that, we will use **add\_PredictServicer\_to\_server** function from (**ML\_example\_pb2\_grpc.py**) file that was generated before to add the class PredictSevicer to the server.



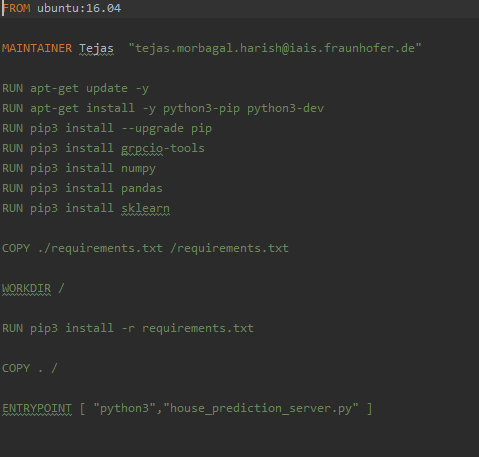
## Step 5: Creating the Client:

In the client file we will do the following:

* Open a gRPC channel
* Create a [stub](https://whatis.techtarget.com/definition/stub)
* Create a request message
* Use the stub to call the service



Step 6: Prepare the Docker file



Build the docker image

1. docker build -t ml\_regression\_grpc:latest .

Run the docker image

2) docker run --name regression\_grpc -p50051:50051 ml\_regression\_grpc:latest

Open one more terminal and run the client which now can access the docker server

3) python3 house\_prediction\_client.py