Deployment of model and run the code

We built a model using Pandas, Sci-kit Learn, and Jupyter Notebooks. In order to change lives and make an impact in the world, the model needs to be deployed in a way it's usable by the general public and not just by the domain experts.

To deploy a machine learning models as a general API over the web that can be used over the various platforms such as websites, Android or IOS apps, Alex skills etc.

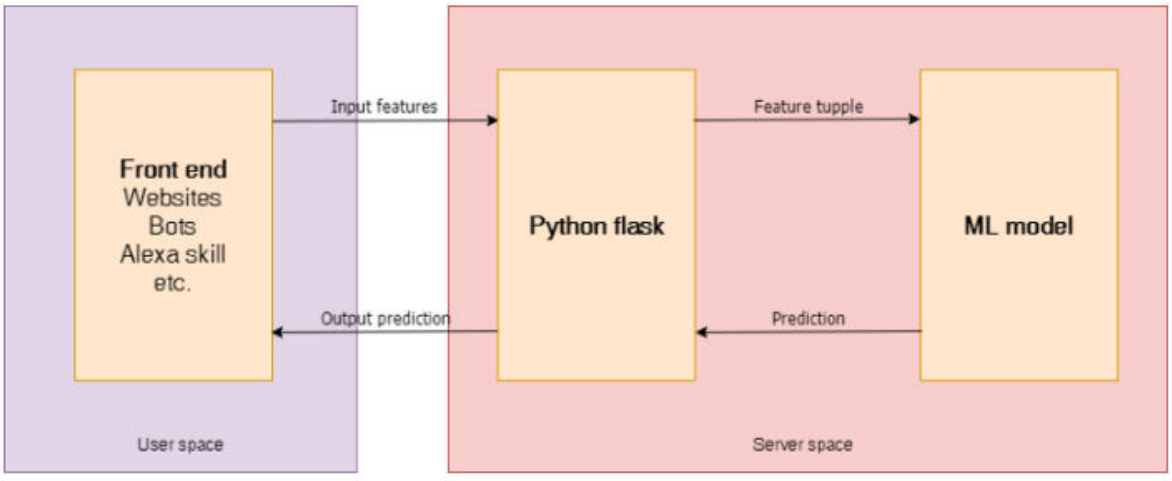
Deployment of Python code

Following are the few libraries and resources which will be used:

1. Pickle: It is a python library to save (serialize) and load (de-serialize) python objects as files on the disk.
2. Flask: It is a python based web framework.
3. pythonanywhere: A free to use educational website that allows hosting python flask and provides a complete python development environment.

Environment Setup:
pip install the flask, flask_cors, jsonify and other python packages.

API Data Flow



API Data flow

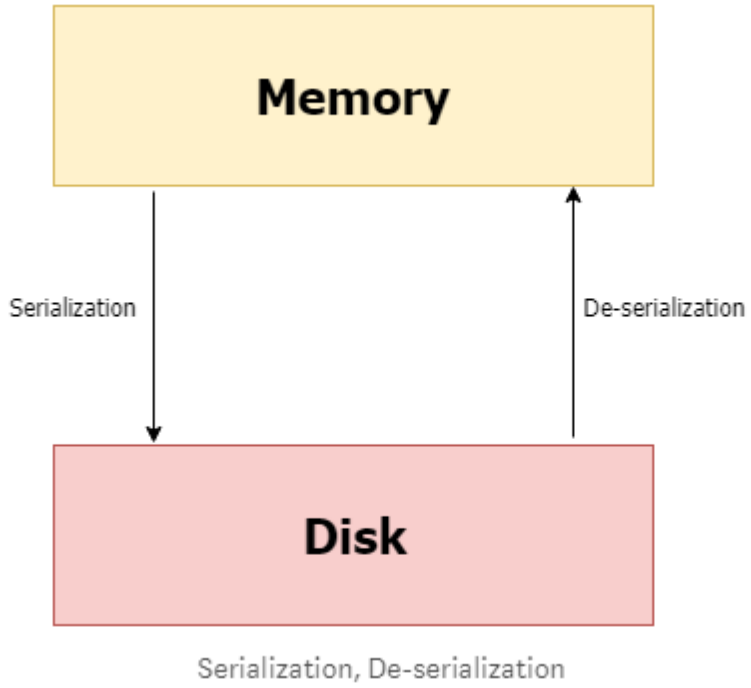
Steps what need to be done as follows,

1. Train the model using Jupyter notebook

2. Save the trained model as a pickle file (Serialization)

3. Create a flask environment that will have an API end point which would encapsulate our trained model and enable it to receive inputs through GET requests over HTTP/HTTPS and then return the output after de-serializing the trained model.
4. Upload the flask script along with trained model on pythonanywhere.

We already trained LightGBM model, so we will do serializing and de-serializing the trained model by using pickle package.



Serializing the trained model as follows,

```
import pickle
from sklearn.model_selection import GridSearchCV

# randomforest is the object we will be serializing,

## Grid Search CV for random Forest model
regr = RandomForestRegressor(random_state = 0) n_estimator = list(range(11,20,1)) depth =
list(range(5,15,2)) # Create the grid grid_search = {'n_estimators': n_estimator, 'max_depth':
depth}

 ## Grid Search Cross-Validation with 5 fold CV
gridcv_rf = GridSearchCV(regr, param_grid = grid_search, cv = 5)
gridcv_rf = gridcv_rf.fit(X_train,y_train)
 view_best_params_GRF = gridcv_rf.best_params_

#Serialize the rf object as a pickle file rf.pkl
```

```
with open('rf.pkl','wb') as handle:
pickle.dump(rf,handle,pickle.HIGHEST_PROTOCAL)
# we used wb to open output as binary and use a higher pickling protocol.
```

After running above code, pickle file created as "rf.pkl' and which is trained model that can be transferred anywhere.

De-serializing the trained model as follows,

```
import pickle
from sklearn.model_selection import GridSearchCV

# rf is the object we will be de-serializing,
#de-serialize the rf object as a pickle file rf.pkl
with open('rf.pkl','rb') as handle:
rf=pickle.load(handle)
#then used to predict the model using test data
predictions_GRF = gridcv_rf.predict(X_test)
```

Flask setup:
Let us setup flask server on the local host and then deploy it on pythonanywhere free

```
import pickle
import json
import os
from flask import Flask,jsonify, request
from flask_cors import CORS
from sklearn.ensemble import RandomForestRegressor
app=Flask(_name_)
CORS(app)
@app.route("/predictions_GRF /",methods=['GET'])
def return_ predictions_GRF ():
X_test=requests.args.get('test')

predictions_GRF =gridcv_rf.predict(X_test)

pred_dict={ "model":"rf",

"pred":" predictions_GRF"

}

return jsonify(pred_dict)

@app.route("/",methods=['GET'])

def default():
```

```
return “<h> Welcome to Cab Fare Prediction<h>”

if __name__==”__main__”:

app.run()
```

- **App route**- App route decorator is used for specifying the flask app route over the web
- **Arguments-** request.args.get(‘test_data’)
- **flask.jsonify()-** It will return python dictionary as JSON.
- **CORS-** It is used to fix CORS headers used for AJAX calls making cross-origin AJAX possible.

After creating the flask environment**,** upload the flask script along with trained model on pythonanywhere free for production.

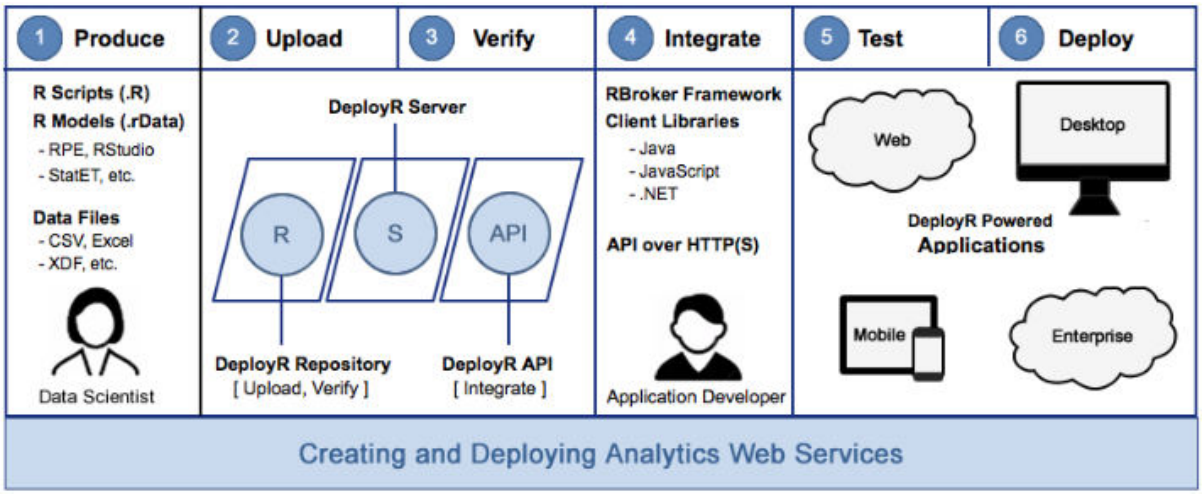
Deployment of R code

Here we are using DeployR to demonstrate and plumber r package used to deploy the model.

DeployR

It is an integration technology for deploying R analytics inside web, desktop, mobile and dashboard applications as well as backend systems.

Basic work flow of DeployR



The above diagram captures the basic work flow used by data scientists and application developers when collaborating on the delivery of solutions powered by Analytics web services. A data scientist develops an R script using standard R tools and publishes that script to deploy server, where it becomes available for execution as a analytics web service. Once published R scripts can be executed by any authorized application using DeployR.

Plumber

Plumber an R package that convert existing R code to a web API by using a handful of special one line comments.

```
#install the plumber
Install.packages(“plumber”)
#import the plumber
library(plumber)
r <- plumb("plumber.R") # Where 'plumber.R' is the location of the file where R code is strored.
#convert to web API
r$run(port=8000) # local host server
```