Deployment:

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 etc

Software Development is all of the activities that make a software system available for the client use.

There are two types of deployment:

1: Online

2: Offline

If the deployment is Online then there will be some kind of User interface to interact.

The tools which can be used for online method are

a: RevoDeployR

b: Orange

c: MS azure.

In this the given user with no coding background could interact and generate the output, by interacting with the simple UI.

The orange and the RevodeployR would provide you the framework and the given data needs to be run over the cloud. It would not store in the client or the local premises.

But this is a drawback as the data in the Banking and the Finance industries prefer offline due to the data security. This method is used for Market researcher and E-commerce

The offline methods are:

a: Schedulers: this is used to schedule a time like weekly or daily and it would automatically run the script in the background and the output is generated.

b: Programming language (Java, python, C, VB)

There are certain libraries which help to connect the model with the given language and then deploy the model.

c: Database or SQL script

d: PMML: This another way to deploy the R and python code and in this it stores the code as a wrapper and then it is deployed.

Flask:

Easy to use.

Built in development server and debugger.

Integrated unit testing support.

RESTful request dispatching.

Extensively documented.

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. python anywhere: A free to use educational website that allows hosting python flask and provides a complete python development environment

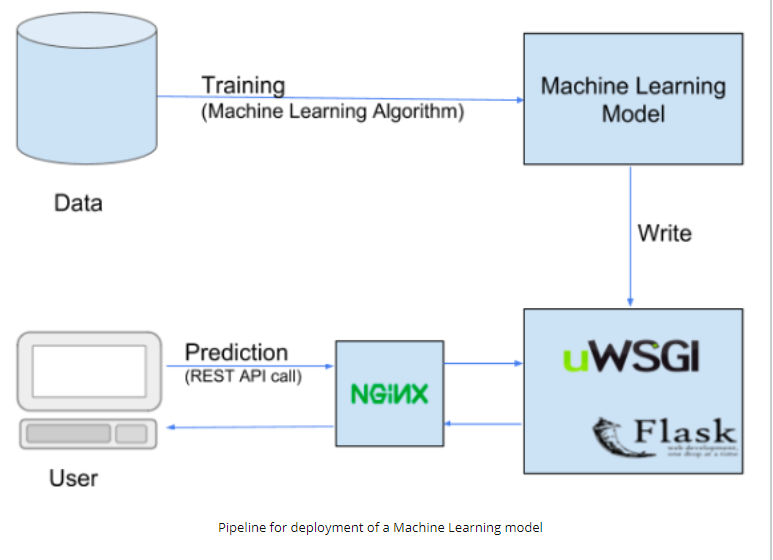
This project has four parts:

model.py — This contains code for the machine learning model to predict sales in the third month based on the sales in the first two months.

app.py — This contains Flask APIs that receives sales details through GUI or API calls, computes the predicted value based on our model and returns it.

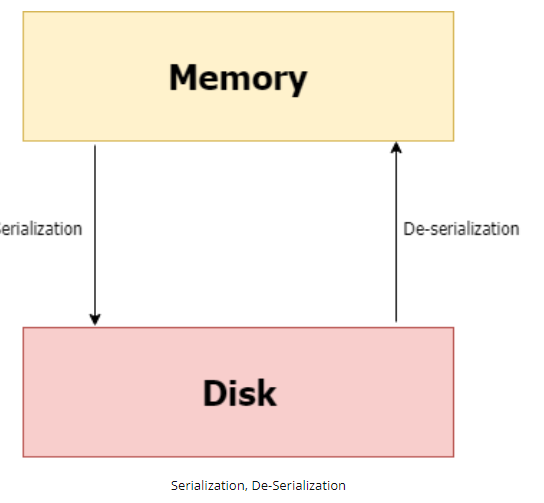
request.py — This uses requests module to call APIs defined in app.py and displays the returned value.

HTML/CSS — This contains the HTML template and CSS styling to allow user to enter sales detail and displays the predicted sales in the third month.



Serializing/De-Serializing

In simple words serializing is a way to write a python object on the disk that can be transferred anywhere and later de-serialized (read) back by a python script.



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.

Code Python:

Serializing the trained model as follows: -

import pickle

import lightgbm as lgb

# lgbm is the object we will be serializing,

#Our trained model

#training data

lgb\_train=lgb.Dataset(X\_train,label=y\_train)

#validation data

lgb\_valid=lgb.Dataset(X\_valid,label=y\_valid)

num\_rounds=10000

lgbm= lgb. train(params, lgb\_train,num\_rounds,valid\_sets=[lgb\_train,lgb\_valid],

ver bose\_eval=1000, early\_stopping\_rounds = 5000)

#Serialize the lgbm object as a pickle file lgbm.pkl

with open(‘lgbm.pkl’,’wb’) as handle:

pickle.dump(lgbm,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 “lgbm.pkl’ and which is trained model that can be transferred anywhere.

De-serializing the trained model as follows:

import pickle

import lightgbm as lgb

# lgbm is the object we will be de-serializing,

#de-serialize the lgbm object as a pickle file lgbm.pkl

with open(‘lgbm.pkl’,’rb’) as handle:

lgbm=pickle.load(handle)

#then used to predict the model using test data

lgbm\_pred=lgbm.predict(X\_test)

**Flask setup:**

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

#pip install "flask"

#pip install "flask-cors"

import pickle

import json

import os

from flask import Flask,jsonify, request

from flask\_cors import CORS

import lightgbm as lgb

app=Flask('\_name\_')

CORS(app)

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

def return\_lgbm\_pred():

X\_test= requests.args.get('test\_data')

lgbm\_pred=lgbm.predict(X\_test)

pred\_dict={ "model":"lgbm","pred":"lgbm\_pred" }

return jsonify(pred\_dict)

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

def default():

return ("<h> Welcome to Santander customer transaction predictor <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.

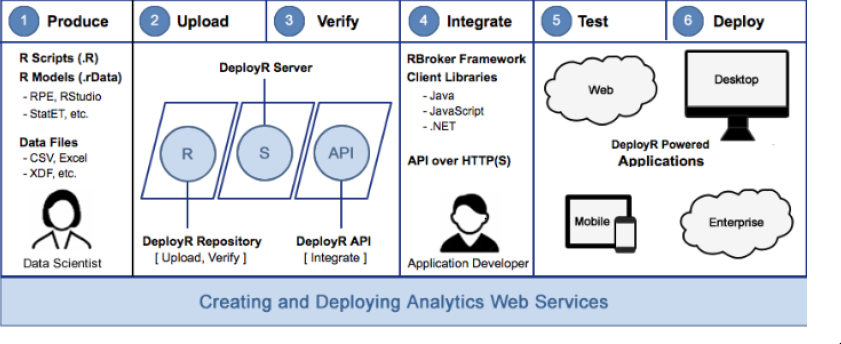
After creating the flask environment, upload the flask script along with trained model on pythonanywhere free for production.

**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



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 stored.

#convert to web API

r**$**run(port**=**8000) # local host server