ML-Model-Flask-Deployment

This is a demo project to elaborate how Machine Learn Models are deployed on production using Flask API

Project Structure

This project has four major parts :

- model.py - This contains code fot our Machine Learning model to predict employee salaries absed on trainign data in 'hiring.csv' file.
- app.py - This contains Flask APIs that receives employee details through GUI or API calls, computes the precited value based on our model and returns it.
- request.py - This uses requests module to call APIs already defined in app.py and dispalys the returned value.
- templates - This folder contains the HTML template to allow user to enter employee detail and displays the predicted employee salary.

jupyter model.py a few seconds ago

File Edit View Language

```python
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import pickle

dataset = pd.read_csv('hiring.csv')

dataset['experience'].fillna(0, inplace=True)

dataset['test_score'].fillna(dataset['test_score'].mean(), inplace=True)

X = dataset.iloc[:, :3]

#Converting words to integer values
def convert_to_int(word):
    word_dict = {'one':1, 'two':2, 'three':3, 'four':4, 'five':5, 'six':6, 'seven':7, 'eight':8,
                 'nine':9, 'ten':10, 'eleven':11, 'twelve':12, 'zero':0, 0: 0}
    return word_dict[word]

X['experience'] = X['experience'].apply(lambda x : convert_to_int(x))

y = dataset.iloc[:, -1]

#Splitting Training and Test Set
#Since we have a very small dataset, we will train our model with all availabe data.

from sklearn.linear_model import LinearRegression
regressor = LinearRegression()

#Fitting model with trainig data
regressor.fit(X, y)

# Saving model to disk
pickle.dump(regressor, open('model.pkl','wb'))

# Loading model to compare the results
model = pickle.load(open('model.pkl','rb'))
print(model.predict([[2, 9, 6]]))
```

jupyter hiring.csv a few seconds ago

File Edit View Language

```
experience,test_score,interview_score,salary
,8,9,50000
,8,6,45000
five,6,7,60000
two,10,10,65000
seven,9,6,70000
three,7,10,62000
ten,,7,72000
eleven,7,8,80000
```

jupyter app.py ✔ a few seconds ago

File Edit View Language

```python
import numpy as np
from flask import Flask, request, jsonify, render_template
import pickle

app = Flask(__name__)
model = pickle.load(open('model.pkl', 'rb'))

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/predict',methods=['POST'])
def predict():
    '''
    For rendering results on HTML GUI
    '''
    int_features = [int(x) for x in request.form.values()]
    final_features = [np.array(int_features)]
    prediction = model.predict(final_features)

    output = round(prediction[0], 2)

    return render_template('index.html', prediction_text='Employee Salary should be $ {}'.format(output))

@app.route('/predict_api',methods=['POST'])
def predict_api():
    '''
    For direct API calls trought request
    '''
    data = request.get_json(force=True)
    prediction = model.predict([np.array(list(data.values()))])

    output = prediction[0]
    return jsonify(output)

if __name__ == "__main__":
    app.run(debug=True)
```

jupyter request.py ✔ a few seconds ago

File Edit View Language

```python
import requests

url = 'http://localhost:5000/predict_api'
r = requests.post(url,json={'experience':2, 'test_score':9, 'interview_score':6})

print(r.json())
```

Create the machine learning model, run model.py

jupyter Model.py (autosaved)

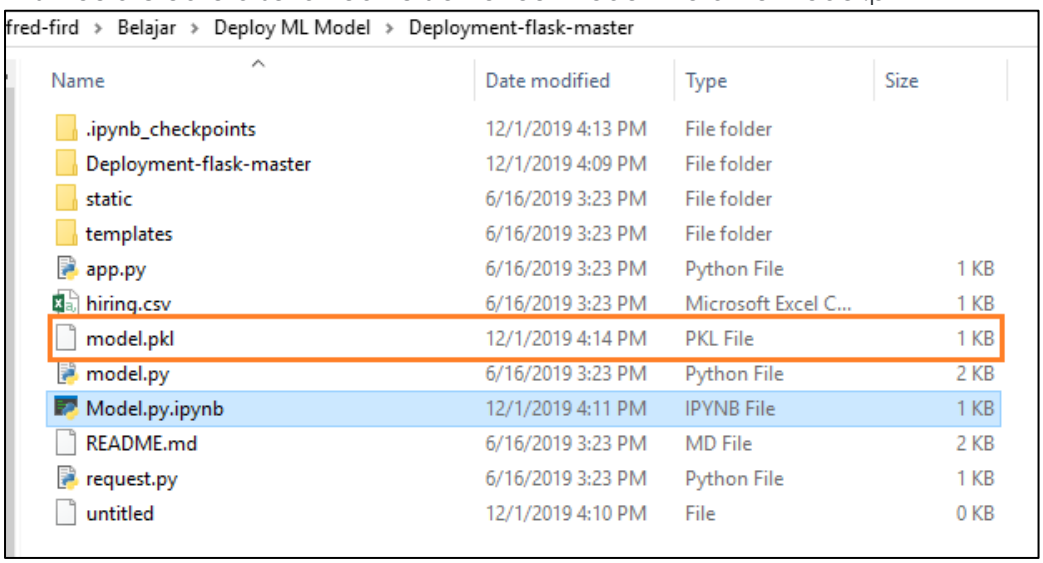
File Edit View Insert Cell Kernel Widgets Help

Run Code

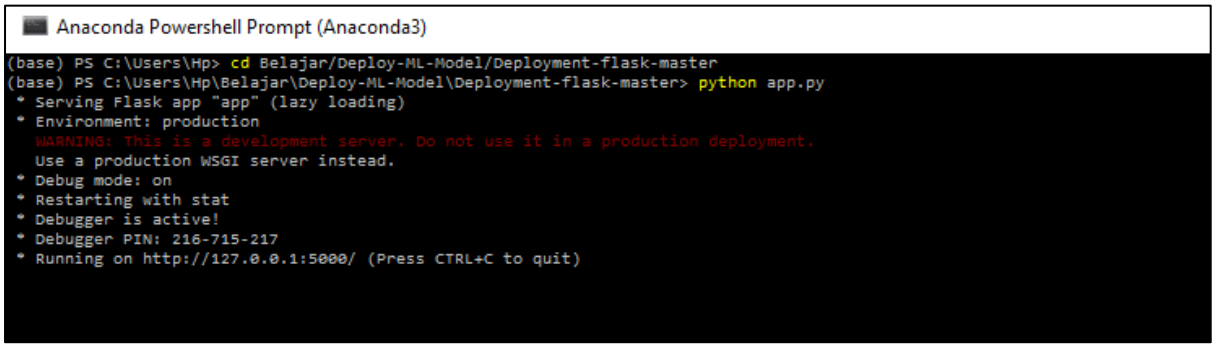
In [1]:
```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import pickle

dataset = pd.read_csv('hiring.csv')

dataset['experience'].fillna(0, inplace=True)

dataset['test_score'].fillna(dataset['test_score'].mean(), inplace=True)

X = dataset.iloc[:, :3]

#Converting words to integer values
def convert_to_int(word):
    word_dict = {'one':1, 'two':2, 'three':3, 'four':4, 'five':5, 'six':6, 'seven':7, 'eight':8,
                 'nine':9, 'ten':10, 'eleven':11, 'twelve':12, 'zero':0, 0: 0}
    return word_dict[word]

X['experience'] = X['experience'].apply(lambda x : convert_to_int(x))

y = dataset.iloc[:, -1]

#Splitting Training and Test Set
#Since we have a very small dataset, we will train our model with all availabe data.

from sklearn.linear_model import LinearRegression
regressor = LinearRegression()

#Fitting model with trainig data
regressor.fit(X, y)

# Saving model to disk
pickle.dump(regressor, open('model.pkl','wb'))

# Loading model to compare the results
model = pickle.load(open('model.pkl','rb'))
print(model.predict([[2, 9, 6]]))
```

```
[53290.89255945]
```

This would create a serialized version of our model into a file model.pkl



Run app.py using 'python app.py' command to start Flask API



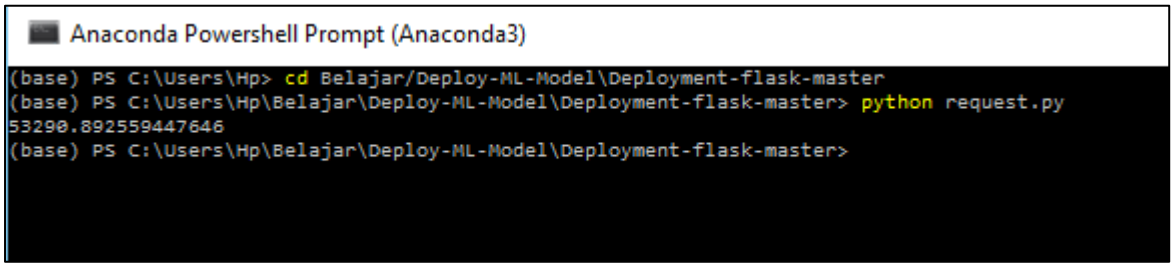
By default, flask will run on port 5000.

Navigate to URL http://localhost:5000
You should be able to view the homepage as below :

Enter valid numerical values in all 3 input boxes and hit Predict.

Send direct POST requests to FLask API using Python's inbuilt request module

Run python request.py command to send the request with some pre-population values.



Hosting the flask app

flask app hosted on the localhost can't be shared with others since it's "local". Host the script to a free python hosting website called pythonanywhere by following a few simple steps:

- Sign in for pythonanywhere account.
- Add a new web app

  I Choose flask and python version 3.7. After creating the web app, we will get a URL that points to flask endpoint. By default, [username].pythonanywhere.com. My URL: fredfird.pythonanywhere.com
- Install dependencies

  this ML model using various external libraries such as sklearn, numpy, pandas etc. Install all of them in your pythonanywhere environment. To do so, open a new bash console and install libraries using pip. Make sure to install libraries with pip using the user option because we don't get the superuser rights.

  ```
  pip install flask flask_cors jsonify numpy pandas
  ```

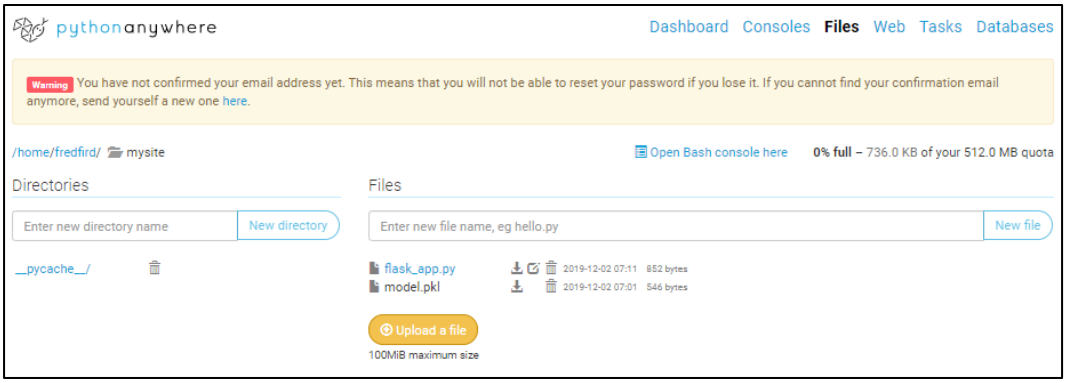
```
  Downloading https://files.pythonhosted.org/packages/63/5d/44962520aa458852bbd40ac5553dd1431f6367ab31f39ee7650b57136d9a/jsonify-0.5.tar.gz
Building wheels for collected packages: jsonify
  Building wheel for jsonify (setup.py) ... done
  Created wheel for jsonify: filename=jsonify-0.5-cp27-none-any.whl size=1563 sha256=69a95f1de4c1374b8ff739abcb43dca11e7e8c0419612244ed6595623e1f327
4
  Stored in directory: /home/fredfird/.cache/pip/wheels/ba/51/8a/136c19b604769b6cb4946fc3402a7068d86fe1f43f8824e2bf
Successfully built jsonify
Installing collected packages: jsonify
ERROR: Could not install packages due to an EnvironmentError: [Errno 13] Permission denied: '/usr/local/lib/python2.7/dist-packages/jsonify-0.5.dist
-info'
Consider using the `--user` option or check the permissions.

07:06 ~/mysite $ pip install flask flask_cors jsonify numpy pandas
DEPRECATION: Python 2.7 will reach the end of its life on January 1st, 2020. Please upgrade your Python as Python 2.7 won't be maintained after that
 date. A future version of pip will drop support for Python 2.7. More details about Python 2 support in pip, can be found at https://pip.pypa.io/en/
latest/development/release-process/#python-2-support
Looking in links: /usr/share/pip-wheels
Requirement already satisfied: flask in /usr/local/lib/python2.7/dist-packages (1.1.1)
Collecting flask_cors
  Using cached https://files.pythonhosted.org/packages/78/38/e68b11daa5d613e3a91e4bf3da76c94ac9ee0d9cd515af9c1ab80d36f709/Flask_Cors-3.0.8-py2.py3-n
one-any.whl
Processing /home/fredfird/.cache/pip/wheels/ba/51/8a/136c19b604769b6cb4946fc3402a7068d86fe1f43f8824e2bf/jsonify-0.5-cp27-none-any.whl
Requirement already satisfied: numpy in /usr/local/lib/python2.7/dist-packages (1.16.5)
Requirement already satisfied: pandas in /usr/local/lib/python2.7/dist-packages (0.24.2)
Requirement already satisfied: itsdangerous>=0.24 in /usr/local/lib/python2.7/dist-packages (from flask) (1.1.0)
Requirement already satisfied: Jinja2>=2.10.1 in /usr/local/lib/python2.7/dist-packages (from flask) (2.10.3)
Requirement already satisfied: click>=5.1 in /usr/local/lib/python2.7/dist-packages (from flask) (7.0)
Requirement already satisfied: Werkzeug>=0.15 in /usr/local/lib/python2.7/dist-packages (from flask) (0.16.0)
Requirement already satisfied: Six in /usr/local/lib/python2.7/dist-packages (from flask_cors) (1.13.0)
Requirement already satisfied: python-dateutil>=2.5.0 in /usr/local/lib/python2.7/dist-packages (from pandas) (2.8.0)
Requirement already satisfied: pytz>=2011k in /usr/local/lib/python2.7/dist-packages (from pandas) (2019.3)
Requirement already satisfied: MarkupSafe>=0.23 in /usr/lib/python2.7/dist-packages (from Jinja2>=2.10.1->flask) (0.23)
Installing collected packages: flask-cors, jsonify
ERROR: Could not install packages due to an EnvironmentError: [Errno 13] Permission denied: '/usr/local/lib/python2.7/dist-packages/flask_cors'
Consider using the `--user` option or check the permissions.

07:07 ~/mysite $
```

- Upload the files

  Inside the default folder — /mysite/ you need to upload your complete folder. You can do it either using the files page on the website or using the bash console by using wget command to download your files.

/home/fredfird/mysite/flask_app.py

```python
# Create API of ML model using flask

'''
This code takes the JSON data while POST request an performs the prediction using loaded model and returns
the results in JSON format.
'''

# Import libraries
from flask import Flask, request, jsonify
from sklearn.externals import joblib
import pandas as pd

app = Flask(__name__)

# Load the model
model = joblib.load(open('/home/fredfird/mysite/model.pkl','rb'))

@app.route('/api/',methods=['POST'])
def predict():
    # Get the data from the POST request.
    data = request.get_json(force=True)

    # convert data into dataframe
    data.update((x, [y]) for x, y in data.items())
    data_df = pd.DataFrame.from_dict(data)

    # predictions
    prediction = model.predict(data_df)

    # Take the first value of prediction
    output = {'prediction': int(prediction[0])}

    return jsonify(output)
```



- Reload the web app
  endpoint will now act as an API to facilitate other applications.