National University of Computer and Emerging Sciences



Lab Manual 07

CL461-Artificial Intelligence Lab

|  |  |
| --- | --- |
| Course Instructor | Dr. Mubasher Baig |
| Lab Instructor (s) | Mahmood Hussain  Saad Ali |
| Section | A |
| Semester | Spring 2021 |

Department of Computer Science

FAST-NU, Lahore, Pakistan

Table of Contents

[Objectives](#_heading=h.30j0zll) **3**

[Task Distribution](#_heading=h.1fob9te) **3**

[**3. Model Deployment**](#_heading=h.rbmnkm7h0pyn) **3**

[3.1 Flask](#_heading=h.pr3m8hxlwyot) 3

[**4. Building a Model**](#_heading=h.do43cuuzkatj) **4**

[4.1 Import the Libraries](#_heading=h.5z7barqmc45d) 4

[4.2 Import the data](#_heading=h.2b10ncptfpnd) 4

[4.3 Features and labels selection](#_heading=h.ckrvv29yibki) 5

[4.4 Train-Test Splits](#_heading=h.tu0drtivf19x) 5

[4.5 Apply the Model](#_heading=h.m747s8l4tptz) 5

[**5. Model Export**](#_heading=h.5tc5y5x7pi77) **6**

[5.1 Pickle](#_heading=h.lnpgf9iuokdb) 6

[5.2 Saving the Model](#_heading=h.vr8rguqxjzv6) 6

[5.3 Load the Model](#_heading=h.tuv8usy2lnic) 6

[5.4 Model Prediction](#_heading=h.c4pnn1rcubnq) 7

[**6. Model Deployment**](#_heading=h.4dkh6ni83d4k) **7**

[6.1 Flask Installation](#_heading=h.7ioakdm3o4vi) 7

[6.2 Import the Libraries](#_heading=h.cuz3uo2kg0ik) 7

[6.3 Loading the saved model](#_heading=h.h0ht3f2jmz03) 7

[6.4 Redirecting the API to the home page index.html](#_heading=h.bmpnuwh7tg3t) 8

[6.5. Redirecting the API to predict the result (Price)](#_heading=h.fdxg2q72wxun) 8

[6.6 Index.html](#_heading=h.ayypcnwg4oou) 9

[6.7. Starting the flask server](#_heading=h.8j2qj3p94oim) 9

[**7. Project Structure**](#_heading=h.d2ekrqyg9o13) **10**

[**8. Running the Project**](#_heading=h.lcu5q5tv3t9w) **10**

[**9. Exercise**](#_heading=h.2pm68yx8nyl5) **11**

[9.1 Deploy titanic survival predictor (Classification)](#_heading=h.qc0mxkejay73)

[9.2 Deploy USD-PKR Currency Exchange Model (Regression)](#_heading=h.idf7sndbsnzk)  **11**

[**10. Submission Instructions:**](#_heading=h.s1twrzd98wq) **11**

# Objectives

After performing this lab, students shall be able to understand machine learning model deployment using flask.

# Task Distribution

|  |  |
| --- | --- |
| **Total Time** | **170 Minutes** |
| Building a Model | 30 Minutes |
| Flask | 20 Minutes |
| Model Deployment | 20 Minutes |
| Exercise | 90 Minutes |
| Online Submission | 10 Minutes |
|  |  |

# 3. Model Deployment

Model Deployment can be defined as the model that is kept in a production environment or a server where it takes input from the user and gives output in real-time. Suppose you have to build a model that predicts whether to approve a loan for a customer or not. The model is trained on features like salary, dependents, loan amount and several other features then in real-time the model will be able to make predictions when you give input of these fields to the model. You have to give entries of features on which the model is trained then only it would be able to make predictions.

## 3.1 Flask

Flask is a micro framework built in Python, which means it provides various tools and libraries for building web applications.

Flask is a web application framework written in Python. It has multiple modules that make it easier for a web developer to write applications without having to worry about the details like protocol management, thread management, etc.

Flask gives a variety of choices for developing web applications and it gives us the necessary tools and libraries that allow us to build a web application.

# 4. Building a Model

We need a pre-trained model first. For this purpose, We are going to build a house pricing prediction system. There is a famous dataset called USA Housing data, Using this data we can estimate a house price using *average area income, number of rooms, number of bedrooms, average age of the house* and *area population*.

The predicted variable is Price. Price is a continuous variable and we need a regression model to forecast its value and here comes the K-Neighbor Regression Model for our rescue.

First step would be to import required libraries. We created a file *model\_income.py*

## 4.1 Import the Libraries

import pandas as pd

import numpy as np

# to split the data

from sklearn.model\_selection import train\_test\_split

# to imply model

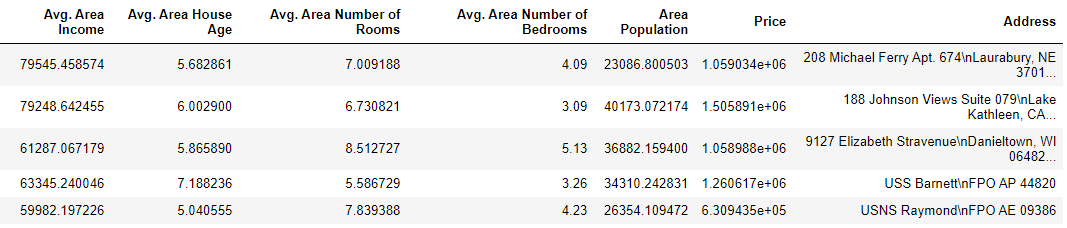
from sklearn.neighbors import KNeighborsRegressor

## 4.2 Import the data

Now, we need to import the data. Pandas is to be used for this purpose.

df=pd.read\_csv('USA\_Housing.csv')

which is:



## 4.3 Features and labels selection

After confirmation, neither null values are present in the dataset nor feature conversion is required. It is required to split the dataset into features and labels. Features are called input columns while labels are the column to be predicted by the model. In our example, *Price* is the label column while all other columns other then *Address* are to be considered as features.

### divide into labels and features ###

x=df.drop(['Price','Address'],axis=1)

y=df['Price']

## 4.4 Train-Test Splits

We are required to make training-testing modules of the dataset.

X\_train,X\_test,Y\_train,Y\_test=train\_test\_split(x,y,test\_size=0.3,random\_state=101)

## 4.5 Apply the Model

Now, its time to import the model and apply it on training dataset.

### KNN Regressor ###

#Import the Model

knn=KNeighborsRegressor()

### Model Training

knn.fit(X\_train,Y\_train)

# 5. Model Export

After building a model, before deploying it is mandatory to save the model first. *Pickle* library is used for this purpose.

## 5.1 Pickle

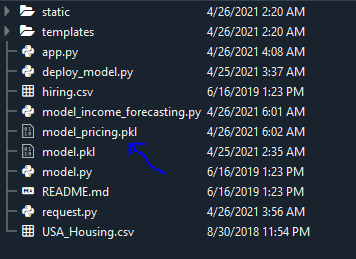
Python pickle module is used for serializing and de-serializing a Python object structure. Any object in Python can be pickled so that it can be saved on disk. What pickle does is that it “serializes” the object first before writing it to file. Pickling is a way to convert a python object (list, dict, etc.) into a character stream. The idea is that this character stream contains all the information necessary to reconstruct the object in another python script.

## 5.2 Saving the Model

To save the model, first it is required to import the pickle library. Lets import the Pickle first.

import pickle

To save model to the disk, *dump* function is used with name and set of permissions.



# Saving model to disk

pickle.dump(knn, open('model\_pricing.pkl','wb'))

Now the model is saved.

## 5.3 Load the Model

To load the model, *load* function is called from the pickle module.

# Loading model to compare the results

model = pickle.load(open('model\_pricing.pkl','rb'))

## 5.4 Model Prediction

To generate the predictions from the model, *predict* is to be used as a model.

print(model.predict([[20000, 5, 7,3,10000]]))

# 6. Model Deployment

Now comes to Model deployment. First of all, we need to setup Flask.

## 6.1 Flask Installation

To install the flask, run the following commands in terminal

pip install flask

## 6.2 Import the Libraries

import numpy as np

from flask import Flask, request, jsonify, render\_template

import pickle

## 6.3 Loading the saved model

We load the *model\_pricing.pkl* file and initialize the flask app.

app = Flask(\_\_name\_\_)

model = pickle.load(open('model\_pricing.pkl', 'rb'))

## 6.4 Redirecting the API to the home page index.html

After initializing the app, we have to tell Flask what we want to do when the web page loads. The line @app.route("/", methods = ["GET","POST"]) tells Flask what to do when we load the home page of website.

@app.route('/')

def home():

return render\_template('index.html')

We use **@app.route(‘/’)** to define functions which are used to redirect them into any number of URI with respect to the API. So, when you start the flask server, it redirects to index.html file by default in our case.

## 6.5. Redirecting the API to predict the result (Price)

@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)]

final\_features=np.reshape(final\_features, (-1, 1)).T

prediction = model.predict(final\_features)

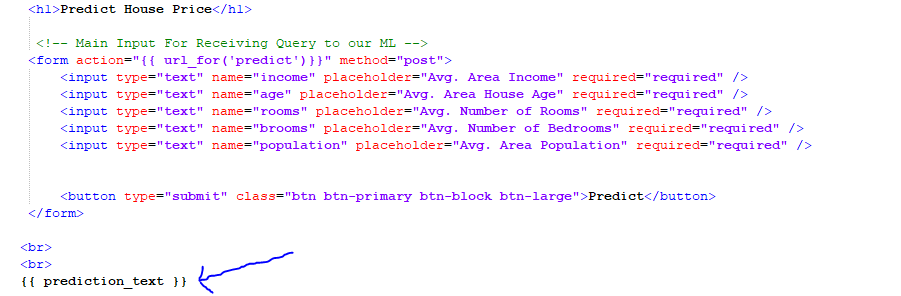
output = round(prediction[0], 2)

return render\_template('index.html', prediction\_text='House Price should be $ {}'.format(output))

Since it is a **‘POST’** request, it will be reading the input values from request.form.values(). Now that we have the input values in the variable **int\_features**, we will convert it into an array and then use the model to predict it and round the final prediction to two decimal places.

When we click on the predict button in index.html, it predicts the salary for the values entered by the user (3 inputs), then passes on the variable ``` **output** ``` outputted from the model and sends it back to index.html template as **prediction\_text**.

## 6.6 Index.html



The **{{ prediction\_text }}** placeholder you see here is where your output prediction(salary predicted) from the model will be placed in our index.html file.

## 6.7. Starting the flask server

This will call app.run() and run our web page locally, hosted on your computer.

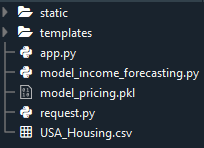
if \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True)

The following script starts the flask server on localhost and default port (5000) making <http://127.0.0.1:5000/>

Just paste [**http://127.0.0.1:5000/**](http://127.0.0.1:5000/) (or) [**http://localhost:5000**](http://localhost:5000/) on browser and press enter to see the server working.

# 7. Project Structure



This project has four major parts:

**1. model\_income\_forecasting.py** — This contains code for our Machine Learning model to predict employee salaries based on training data in **‘hiring.csv’** file.

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

**3. template** — This folder contains the HTML template (**index.html**) to allow user to enter employee detail and displays the predicted employee salary.

**4. static** — This folder contains the **css** folder with **style.css** file which has the styling required for out index.html file.

# 8. Running the Project

To run the project,

python app.py

# 9. Exercise

## 9.1 Deploy titanic survival predictor (Classification) (15)

We developed the titanic survival predictor machine learning model during the previous lab. You are required to deploy the model using Flask

## 9.2 Deploy USD-PKR Currency Exchange Model (Regression) (10)

You are required to deploy the Regression Model for currency exchange we developed during the previous lab.

# 10. Submission Instructions:

1. A data file is attached. For Practice Exercise, One has to use this file.
2. For Examples given in manual, no dataset is required because the dataset used is available in sklearn library.
3. To make the submission, Create a Jupyter Notebook File (lab7\_rollno.ipynb), create a .zip file along with the data file and submit on the Portal.