Deployment on Flask

Name: Seoyoung Kim
Batch Code: LISUM25
Date: Nov. 2023
Submitted to: https://github.com/syoungk7/Model_Deployment

1. Dataset: Adult income dataset

https://www.kaggle.com/datasets/wenruliu/adult-income-dataset/

Fields: The dataset contains 16 columns
Target filed: Income -- The income is divided into two classes: <= 50K and > 50K
Number of attributes: 14 -- These are the demographics and other features to describe a person

2. Model used: Decision Tree using DecisionTreeClassifier()

A Decision Tree is a popular machine learning algorithm used for both classification and regression tasks. It works by recursively partitioning the data into subsets based on the most significant attribute at each step.

The key concepts are:

- **Nodes**: Decision Trees consist of nodes, which represent features or attributes in the dataset.
- **Root Node**: The topmost node in a Decision Tree is called the root node. It represents the entire dataset and is split into subsets based on a selected attribute.
- **Decision Nodes** (Internal Nodes): Nodes that follow the root node are decision nodes. They represent a decision or a test on an attribute, leading to different branches.
- **Leaves** (Terminal Nodes): The end nodes of a Decision Tree are called leaves or terminal nodes. They represent the final output or decision, such as a class label in classification or a numerical value in regression.
- **Branches**: The edges connecting nodes represent the outcome of a decision. The tree structure is formed by these branches.
- **Splitting**: At each decision node, the dataset is split into subsets based on a specific feature or attribute. The goal is to create homogeneous subsets that are more predictable.

- **Decision Criteria**: The criteria for splitting nodes are determined based on metrics like Gini impurity (for classification) or mean squared error (for regression). These metrics quantify the purity or homogeneity of the subsets.
- **Pruning**: Decision Trees can be prone to overfitting, capturing noise in the data. Pruning involves removing unnecessary branches to improve the model's generalization to new, unseen data.

3. **Model build**: preprocessing.py

(reference: https://www.geeksforgeeks.org/deploy-machine-learning-model-using-flask/)

```python
import pandas as pd
import numpy as np
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier

# Load dataset
df = pd.read_csv('./dataset/sample.csv')

# replace missing values
c_names = df.columns
for c in c_names:
    df[c] = df[c].replace("?", np.NaN)
df = df.apply(lambda x:x.fillna(x.value_counts().index[0]))

# change column names to simple
df.replace(['Divorced', 'Married-AF-spouse', 'Married-civ-spouse', 'Married-spouse-absent', 'Never-married', 'Separated', 'Widowed'],
           ['divorced', 'married', 'married', 'married','not married', 'not married', 'not married'], inplace=True)

# drop redundant columns
df = df.drop(['fnlwgt', 'educational-num'], axis=1)


# label Encoder
categories = ['workclass', 'race', 'education','marital-status', 'occupation','relationship', 'gender', 'native-country', 'income']
labelEncoder = preprocessing.LabelEncoder()

# map numerical values to categorical labels
mapping_dict = {}
for col in categories:
    df[col] = labelEncoder.fit_transform(df[col])
    le_name_mapping = dict(zip(labelEncoder.classes_, labelEncoder.transform(labelEncoder.classes_)))
    mapping_dict[col] = le_name_mapping
# print(mapping_dict)

# final featured data and labels
X = df.values[:, 0:12]
Y = df.values[:,12]

# split dataset into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3, random_state=100)

# apply model: DecisionTree
dt_clf = DecisionTreeClassifier(criterion = "gini", random_state=100, max_depth=5, min_samples_leaf=5)
dt_clf.fit(X_train, y_train)
y_pred = dt_clf.predict(X_test)
print ("Index\nAccuracy using Desicion Tree is ", accuracy_score(y_test, y_pred) * 100)

#creat model as model.pkl
import pickle
pickle.dump(dt_clf, open("model.pkl","wb"))
```

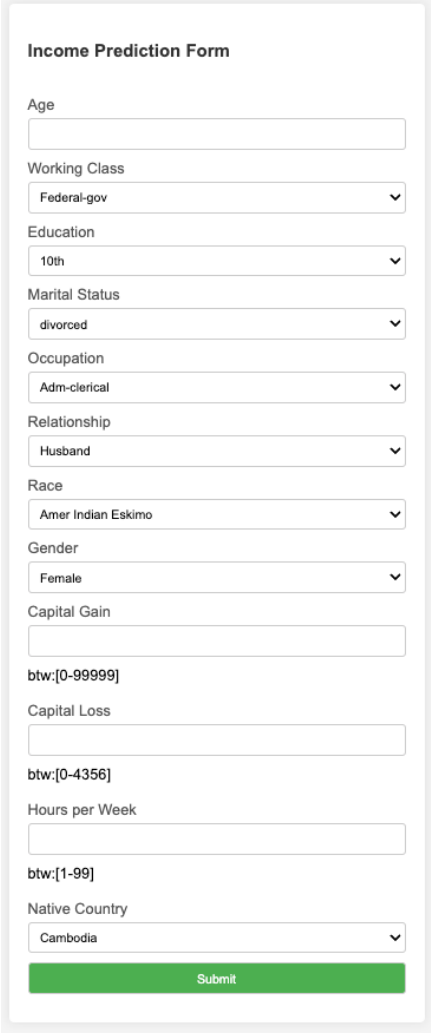
```
(base) syoungk@ees-fvfgg2hvq05p income-prediction % python3 preprocessing.py
Index
Accuracy using Desicion Tree is  83.26622534634545
(base) syoungk@ees-fvfgg2hvq05p income-prediction %
```

4. HTML files for web deployment: index.html and result.html

index.html

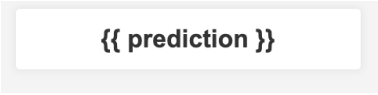
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Income Prediction Form</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f4f4f4;
            margin: 20px;
        }

        h3 {
            color: #333;
        }

        form {
            background-color: #fff;
            padding: 20px;
            border-radius: 5px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
            max-width: 400px;
            margin: auto;
        }

        label {
            display: block;
            margin-bottom: 5px;
            color: #555;
        }

        input,
        select {
            width: 100%;
            padding: 8px;
            margin-bottom: 10px;
            box-sizing: border-box;
            border: 1px solid #ccc;
            border-radius: 4px;
        }

        input[type="submit"] {
            background-color: #4caf50;
            color: #fff;
            cursor: pointer;
        }

        input[type="submit"]:hover {
            background-color: #45a049;
        }
    </style>
```

```
<body>

    <form action="/result" method="POST">
        <h3>Income Prediction Form</h3><br>
        <label for="age">Age</label>
        <input type="text" id="age" name="age"><br>

        <label for="w_class">Working Class</label>
        <select id="w_class" name="w_class">
            <option value="0">Federal-gov</option>
            <option value="1">Local-gov</option>
            <option value="2">Never-worked</option>
            <option value="3">Private</option>
            <option value="4">Self-emp-inc</option>
            <option value="5">Self-emp-not-inc</option>
            <option value="6">State-gov</option>
            <option value="7">Without-pay</option>
        </select><br>

        <label for="edu">Education</label>
        <select id="edu" name="edu">
            <option value="0">10th</option>
            <option value="1">11th</option>
            <option value="2">12th</option>
            <option value="3">1st-4th</option>
            <option value="4">5th-6th</option>
            <option value="5">7th-8th</option>
            <option value="6">9th</option>
            <option value="7">Assoc-acdm</option>
            <option value="8">Assoc-voc</option>
            <option value="9">Bachelors</option>
            <option value="10">Doctorate</option>
            <option value="11">HS-grad</option>
            <option value="12">Masters</option>
            <option value="13">Preschool</option>
            <option value="14">Prof-school</option>
            <option value="15">16 - Some-college</option>
        </select><br>

        <label for="martial_stat">Marital Status</label>
        <select id="martial_stat" name="martial_stat">
            <option value="0">divorced</option>
            <option value="1">married</option>
            <option value="2">not married</option>
        </select><br>

        <label for="occup">Occupation</label>
        <select id="occup" name="occup">
            <option value="0">Adm-clerical</option>
            <option value="1">Armed-Forces</option>
            <option value="2">Craft-repair</option>
            <option value="3">Exec-managerial</option>
```



result.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Income Prediction Result</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f4f4f4;
            margin: 20px;
            text-align: center;
        }

        h1 {
            color: #333;
            background-color: #fff;
            padding: 20px;
            border-radius: 5px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
            max-width: 400px;
            margin: auto;
        }
    </style>
</head>
<body>
    <h1>{{ prediction }}</h1>
</body>
</html>
```



5. Model deployment using Flask

(reference: https://phoenixnap.com/kb/install-flask)

script.py

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

# apply flask
app=Flask(__name__)

# home
@app.route('/')

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

def ValuePredictor(to_predict_list): # for prediction
    to_predict = np.array(to_predict_list).reshape(1, 12)
    loaded_model = pickle.load(open("model.pkl", "rb"))
    result = loaded_model.predict(to_predict)
    return result[0]

# result
@app.route('/result', methods = ['POST'])
def result():
    if request.method == 'POST':
        to_predict_list = request.form.to_dict()
        to_predict_list = list(to_predict_list.values())
        to_predict_list = list(map(int, to_predict_list))
        result = ValuePredictor(to_predict_list)

        if int(result) == 1:
            prediction = 'Your predicted income is more than 50K'
        else:
            prediction = 'Your predicted income is less than 50K'

        return render_template("result.html", prediction=prediction)

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

Step 1: Install virtualenv

pip install virtualenv

Step 2: Create an Environment

1. Make a separate directory for your project <income-prediction>
2. Move into the directory
3. Within the directory, create the virtual environment for Flask. When you create the environment, a new folder appears in your project directory with the environment's name.
4. Create an Environment in Linux and MacOS: python3 -m venv venv

Step 3: Activate the Environment

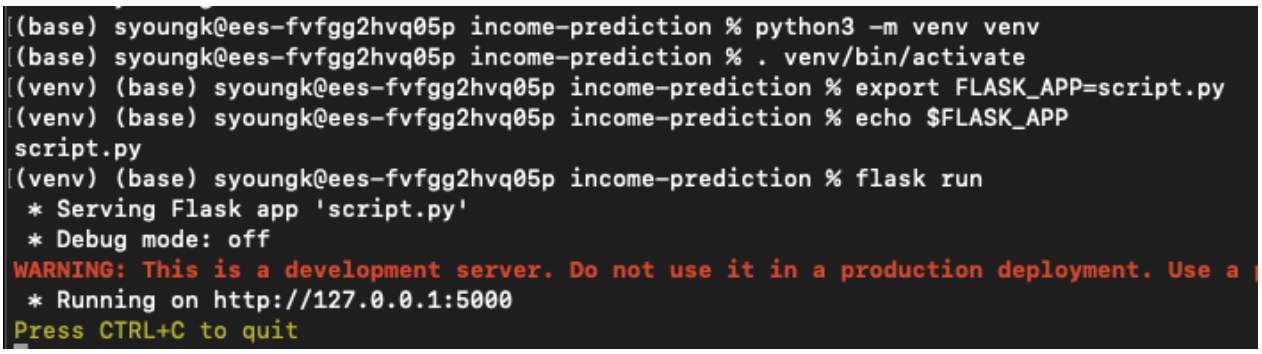
Activate the virtual environment in Linux and MacOS with:. venv/bin/activate

Step 4: Install Flask

Install Flask within the activated environment using pip: pip install Flask

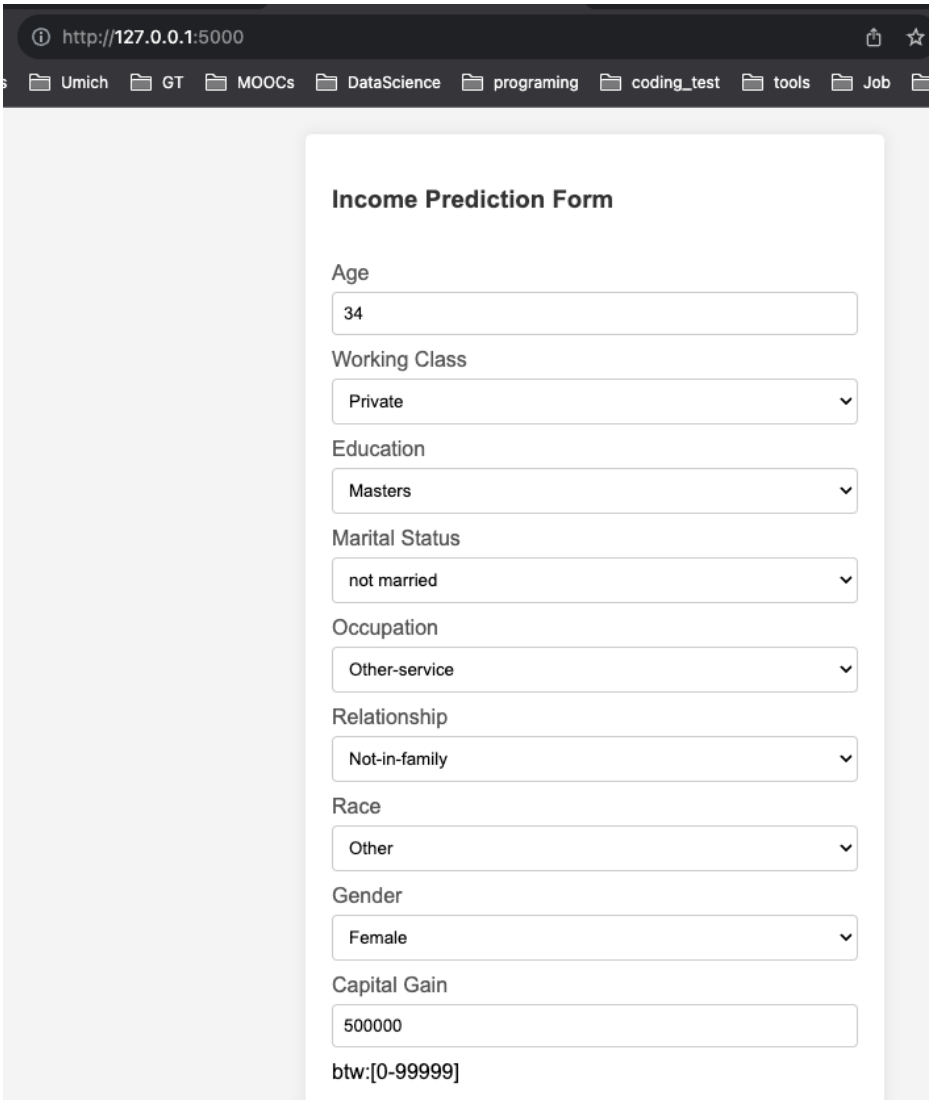
Step 5: Development

1. Using the console, navigate to the project folder using the cd command.
2. Set the FLASK_APP environment variable: export FLASK_APP=script.py
3. Run the Flask application with: flask run
4. Copy and paste the address into the browser to see the project running:



6. Result

Input form



Prediction from Model