

INTRODUCTION

In this project, we are concerned about the deployment of a linear regression model which predicts the employee salary based on experience, test score and interview score. This will be accomplished with *Flask* - a micro web framework written in Python to allow for the building of web applications. We will first build the model and then deploy it to *flask*.

MODEL BUILDING

DATA INFORMATION

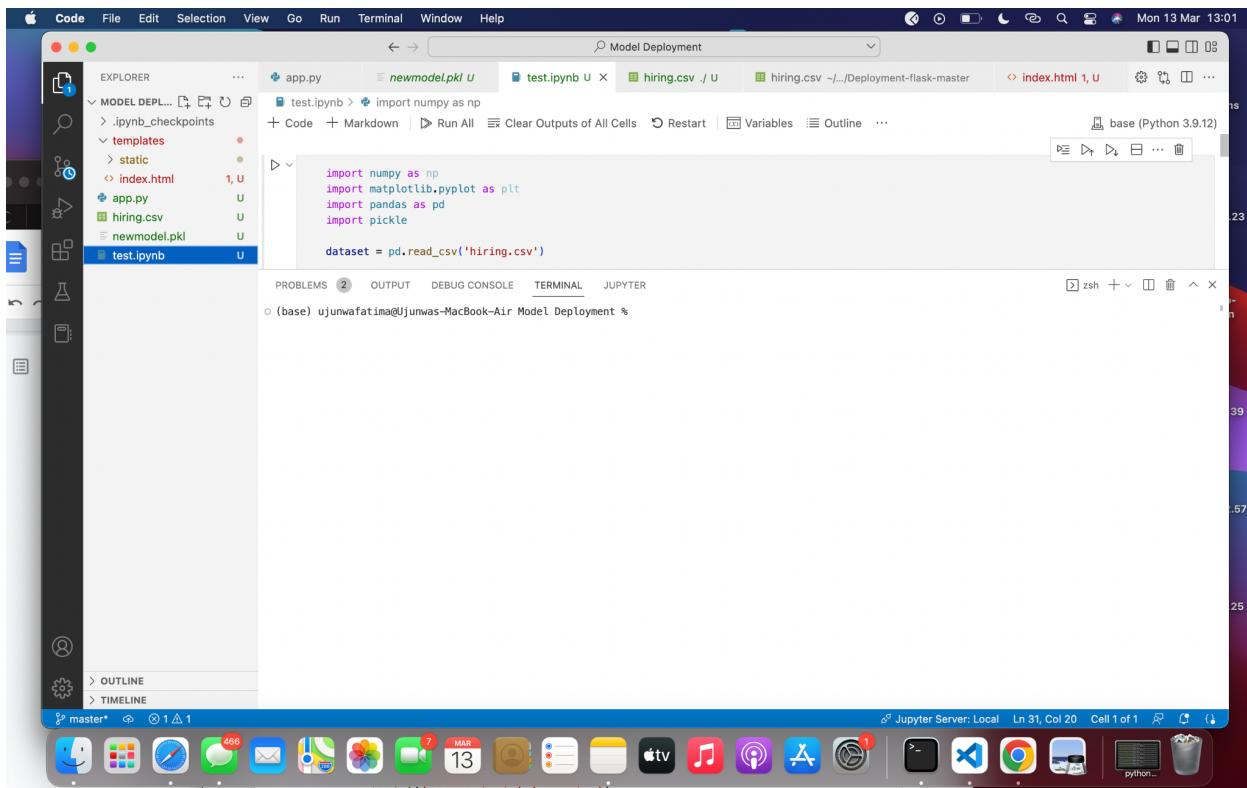
Our data set is an employee data set with 8 rows and 3 columns. The features include experience, *test_score* and *interview_score* while the target variable is the *salary*. Based on the features, we are required to predict the salary of the individual. This is depicted in the table shown below.

	Experience	test_score	interview_score	salary
1	NAN	8	9	50000
2	NAN	8	6	45000
3	five	6	7	60000
4	two	10	10	65000
5	seven	9	6	70000
6	three	7	10	62000
7	ten	NAN	7	72000
8	eleven	7	8	80000

Table 2.1: Dataset Information

IMPORTING LIBRARIES

Here we imported the necessary libraries to aid in our model building as shown in the figure below.



The screenshot shows a Jupyter Notebook interface running on a Mac. The code cell contains the following Python code:

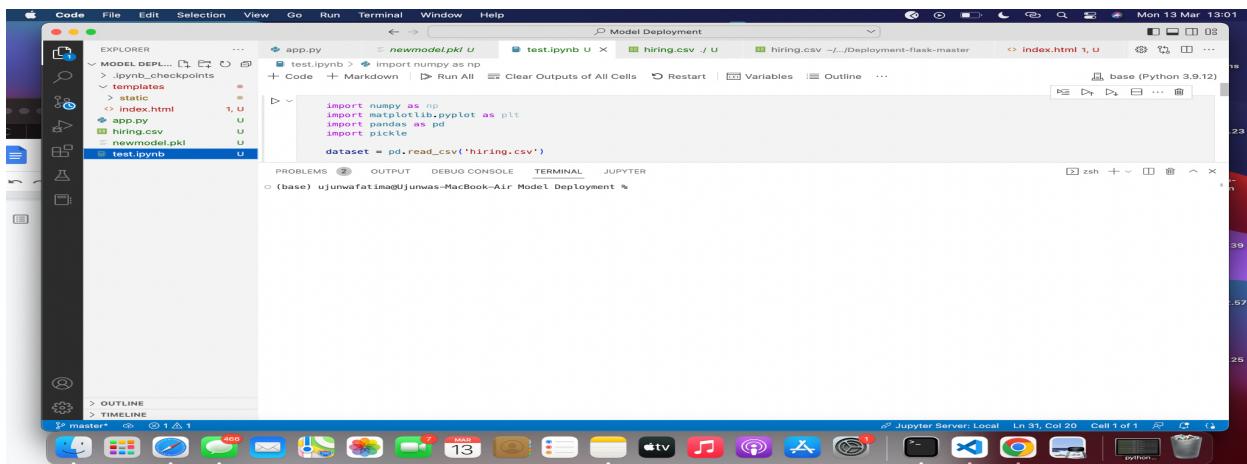
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import pickle

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

The notebook also lists other files in the directory: app.py, newmodel.pkl, hiring.csv, index.html, and test.ipynb. The status bar at the bottom indicates "Jupyter Server: Local Ln 31, Col 20 Cell 1 of 1".

LOADING THE DATA

The dataset is in comma separated value format (CSV) so we used the pandas library to read and convert it into a dataframe.



The screenshot shows a Jupyter Notebook interface running on a Mac. The code cell contains the same Python code as the previous screenshot:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import pickle

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

The notebook lists the same files as before. The status bar at the bottom indicates "Jupyter Server: Local Ln 31, Col 20 Cell 1 of 1".

DATA PREPROCESSING

As we saw in table 2.1, we had a lot of missing values as well as all data points written in words for the *experience* feature. We would be required to convert the data points to int data type and also fill in the missing data to proceed.

```
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]
```

BUILDING THE MODEL

To build the model did not split the data into test and training dataset because of how small our dataset was. We employed the linear regression model to fit our data.

```
#Splitting training and test set
#Since we have a very small dataset, we will train our model with all available data.

from sklearn.linear_model import LinearRegression
regressor = LinearRegression()

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

SAVING THE MODEL TO DISK

We finally saved our model, as ‘newmodel’ using *Pickle*. *Pickle* helps us deploy the model to the Flask framework easily.

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

DEPLOYING MODEL TO FLASK

To achieve this we created a folder *Model Deployment* with the following file arrangement as shown below.

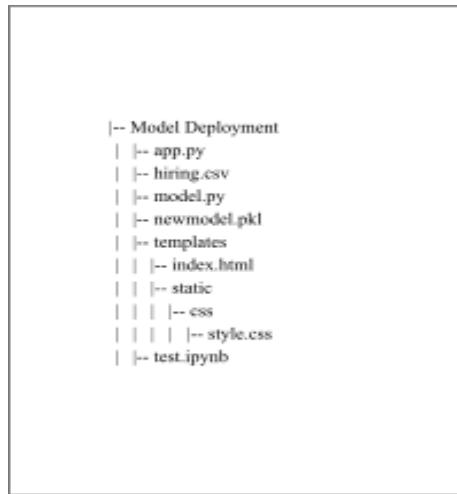


Figure 1.2: Folder Structure

FOLDER STRUCTURE

App.py: This file is to be usually executed by a Python interpreter. We load our save model *newmodel.pkl* and import Flask on this file.

- *@app.route* is used to indicate the endpoint and the method used to execute the function.
- *def predict* is written to save the inputs to variable *int_features*, transform the inputs to array and then finally use the inputs to predict the salary. The return value is then rendered on the html file, *index.html*

The screenshot shows a Jupyter Notebook interface with several open files. The left sidebar includes an 'EXPLORER' section with a 'MODEL DEPLOYMENT' folder containing 'ipyvnb_checkpoints', 'templates', 'static', and 'index.html'. Below these are 'app.py', 'hirings.csv', 'newmodel.pkl', and 'test.ipynb'. The main area displays the content of 'app.py':

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

# TEMPLATE_DIR = os.path.abspath('../templates')
# STATIC_DIR = os.path.abspath('../static')

app = Flask(__name__, static_folder=STATIC_DIR)
model = pickle.load(open('newmodel.pkl', 'rb'))
@app.route('/', methods=['GET'])
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))
if __name__ == "__main__":
    app.run(debug=True, port=4000)
```

Index.html: This file contains the html code that displays the structure of the input fields, buttons and text used on the website. The result of the prediction is also rendered on the web app through this file. We save this file in a folder *templates*.

The screenshot shows a Jupyter Notebook interface with the following details:

- Code Editor:** The main area displays Python code for an application named "Model Deployment". The code includes imports for Flask, numpy, and pandas, along with route definitions for "/predict" and "/".
- File Explorer:** The left sidebar shows the file structure:
 - MODEL DEPL... (selected)
 - lynnb_checkpoints
 - templates
 - index.html (selected)
 - app.py
 - hirings.csv
 - static
 - newmodel.pkl
 - test.ipynb
- Terminal:** At the bottom, the terminal shows the command: `(base) ujjuwanfatima@ujjuwan-MacBook-Air Model Deployment %`.
- Status Bar:** The bottom right corner shows the status bar with information: `Ln 16, Col 25 Tab Size: 4 UTF-8 LF HTML`.

Style.css: This contains the css code applied to give the web application styling. This styling method employed here is the external styling which allows the styling to be done in a single file and linked to the html file. The `.css` file is usually saved in a folder which is in turn saved in a folder named `static`. Flask automatically sources out the styling by looking for this folder.

```

Code File Edit Selection View Go Run Terminal Window Help
... Model Deployment ...
newmodel.pkl U test.ipynb U hiring.csv U hiring.py U index.html 1, U # style.css 9+, U
templates > static > # style.css > # body {
  1 import url(https://fonts.googleapis.com/css?family=Open+Sans);
  2   font-family: "Open Sans", sans-serif;
  3   margin: 12px padding: 10px 4px; margin-bottom: 0; font-size: 13px; line-height: 1.5;
  4   .btn, .btn-primary, .btn-primary:hover, .btn-primary:active, .btn-primary:disabled, .btn[disabled] { background-color: #e6e6e6; border: 1px solid #ccc; border-radius: 5px; color: #333; font-size: 13px; line-height: normal; -webkit-border-radius: 5px; -moz-border-radius: 5px; border: 1px solid #ccc; border-radius: 5px; color: #333; font-size: 13px; line-height: normal; }
  5   .btn-large { padding: 10px 14px; font-size: 15px; line-height: normal; -webkit-border-radius: 5px; -moz-border-radius: 5px; border: 1px solid #ccc; border-radius: 5px; color: #333; font-size: 15px; line-height: normal; }
  6   .btn-primary, .btn-primary:hover { text-shadow: 0 -1px 0 rgba(0, 0, 0, 0.25); color: #fff; }
  7   .btn-primary, .btn-primary:active, .btn-primary:disabled, .btn-primary[disabled] { filter: none; background-color: #4a77d4; border: 1px solid #4a77d4; border-radius: 5px; color: #fff; }
  8   .btn-primary:active, .btn-primary:disabled, .btn-primary[disabled] { background-color: #4a77d4; border: 1px solid #4a77d4; border-radius: 5px; color: #fff; }
  9   .btn-primary:hover { background-color: #4a77d4; border: 1px solid #4a77d4; border-radius: 5px; color: #fff; }
 10   .btn-block { width: 100%; }
 11   .btn-block { width: 100%; height: 100%; overflow: hidden; }
 12   * { -webkit-box-sizing: border-box; -moz-box-sizing: border-box; -ms-box-sizing: border-box; -o-box-sizing: border-box; box-sizing: border-box; }
 13   body { width: 100%; height: 100%; font-family: "Open Sans", sans-serif; background: #092756; color: #fff; font-size: 18px; text-align: center; }
 14   .login { position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%); }
 15   .login form { width: 100%; background: #fff; padding: 10px; border-radius: 5px; }
 16   .login h2 { margin: 0; font-size: 24px; }
 17   .login input { width: 100%; height: 30px; margin-top: 5px; border: 1px solid #ccc; border-radius: 3px; padding: 5px; }
 18   .login button { width: 100%; height: 35px; background: #4a77d4; border: 1px solid #4a77d4; border-radius: 5px; color: #fff; font-size: 16px; margin-top: 10px; }
 19   .login button:hover { background: #4a77d4; border: 1px solid #4a77d4; border-radius: 5px; color: #fff; }
 20   .login button:active { background: #4a77d4; border: 1px solid #4a77d4; border-radius: 5px; color: #fff; }
 21   .login button:disabled { background: #4a77d4; border: 1px solid #4a77d4; border-radius: 5px; color: #fff; }
 22   .login button[disabled] { background: #4a77d4; border: 1px solid #4a77d4; border-radius: 5px; color: #fff; }
 23   .login input:invalid { border: 1px solid #ccc; border-radius: 3px; }
 24   .login input:valid { border: 1px solid #4a77d4; border-radius: 3px; }
 25   .login input:valid ~ button { background: #4a77d4; border: 1px solid #4a77d4; border-radius: 5px; color: #fff; }
 26   .login input:valid ~ button:disabled { background: #4a77d4; border: 1px solid #4a77d4; border-radius: 5px; color: #fff; }
 27   .login input:valid ~ button:disabled { background: #4a77d4; border: 1px solid #4a77d4; border-radius: 5px; color: #fff; }
 28   .login input:valid ~ button:disabled { background: #4a77d4; border: 1px solid #4a77d4; border-radius: 5px; color: #fff; }
 29   .login input:valid ~ button:disabled { background: #4a77d4; border: 1px solid #4a77d4; border-radius: 5px; color: #fff; }
 30   .login input:valid ~ button:disabled { background: #4a77d4; border: 1px solid #4a77d4; border-radius: 5px; color: #fff; }
 31   .login input:valid ~ button:disabled { background: #4a77d4; border: 1px solid #4a77d4; border-radius: 5px; color: #fff; }
 32   .login input:valid ~ button:disabled { background: #4a77d4; border: 1px solid #4a77d4; border-radius: 5px; color: #fff; }
 33   .login input:valid ~ button:disabled { background: #4a77d4; border: 1px solid #4a77d4; border-radius: 5px; color: #fff; }
 34   .login input:valid ~ button:disabled { background: #4a77d4; border: 1px solid #4a77d4; border-radius: 5px; color: #fff; }
PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
(base) ujunwafatima@Ujunwas-MacBook-Air Model Deployment % conda activate base
(base) ujunwafatima@Ujunwas-MacBook-Air Model Deployment % /Users/ujunwafatima/opt/anaconda3/bin/python "/Users/ujunwafatima/Downloads/Model Deployment/app.py"
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Running on http://127.0.0.1:4000/ (Press CTRL+C to quit)
* Restarting with watchdog (fsevents)
* Debugger is active!
* Debugger PIN: 105-966-736

```

RESULTS

Once all the code is written as shown in the figures above, we run the app.py file and load the server. The resulting web page is as shown below:

```

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
(base) ujunwafatima@Ujunwas-MacBook-Air Model Deployment % conda activate base
(base) ujunwafatima@Ujunwas-MacBook-Air Model Deployment % /Users/ujunwafatima/opt/anaconda3/bin/python "/Users/ujunwafatima/Downloads/Model Deployment/app.py"
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Running on http://127.0.0.1:4000/ (Press CTRL+C to quit)
* Restarting with watchdog (fsevents)
* Debugger is active!
* Debugger PIN: 105-966-736

```

Employee Salary should be \$ 30481.77

