

**Week 4:** Deployment on Flask

**Name:** Nishchay Vaid **Batch Code:** LISUM20 **Date:** 27 April 2023 **Submitted to:** Data Glacier

# Table of Contents:

1. [Introduction 3](#_bookmark0)
2. [Data Information 4](#_bookmark1)
   1. [Attribute Information 4](#_TOC_250003)
3. Build Machine Learning Model 5
4. Turning Model into Flask Framework 6
   1. [App.py 7](#_TOC_250002)
   2. Home.html 8
   3. [Style.css 9](#_TOC_250001)
   4. [Running Procedure 10](#_TOC_250000)

2

# Introduction

In this project, we are going to deploying machine learning model (Linear Regression) using the Flask Framework. We are trying to build a machine learning model that can predict the salary of an employee based on his/her years of experience, level of education, and performance score in an interview.

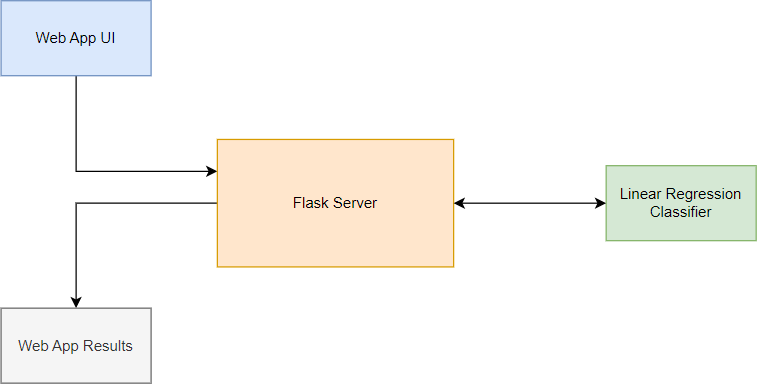


Figure 1.1: Application Workflow

Our main objective is to accomplish two tasks. First, we aim to construct a machine learning model that can be used to analyze the sentiment of YouTube comments. Second, we plan to develop an application programming interface (API) for the model using Flask, which is a lightweight web application framework written in Python. This API will enable us to utilize the predictive capabilities of the model by sending HTTP requests.

3

# Data Information

Table 2.1: Dataset Information

|  |  |  |  |
| --- | --- | --- | --- |
| **experience** | **test\_score** | **interview\_score** | **salary** |
| null | 8 | 9 | 50000 |
| null | 8 | 6 | 45000 |
| five | 6 | 7 | 60000 |
| two | 10 | 10 | 65000 |
| seven | 9 | 6 | 70000 |
| three | 7 | 10 | 62000 |
| ten | null | 7 | 72000 |
| eleven | 7 | 8 | 80000 |

# 2.1 Attribute Information

The collection is composed of one CSV file per dataset, where each line has the following attributes:

Table 2.2: Attribute Information

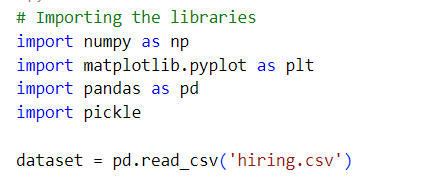
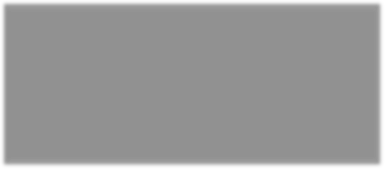
|  |  |
| --- | --- |
| **Attributes** | **Example (1 instance)** |
| experience | two |
| test\_score | 10 |
| interview\_score | 10 |
| salary | 65000 |

4

# Building a Model

## Import Required Libraries and Dataset

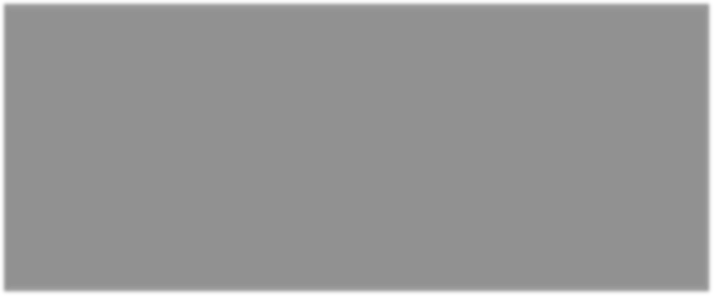
The code below imports necessary libraries such as NumPy, Matplotlib, Pandas, and Pickle. Then it reads a CSV file 'hiring.csv' and loads the data into a pandas dataframe called 'dataset'.



## Data Preprocessing

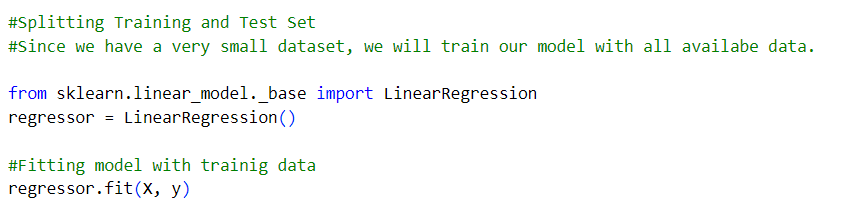
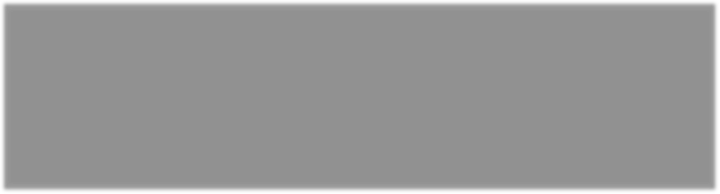
The code performs data preprocessing on a dataset containing information about job applicants. It fills missing values in the 'experience' column with zeros and missing values in the 'test\_score' column with the mean score. The 'experience' column is then converted from string to integer values using a dictionary and a conversion function. Finally, the input and output variables for the machine learning model are created, with the input variables being the first three columns of the dataset and the output variable being the last column.

5



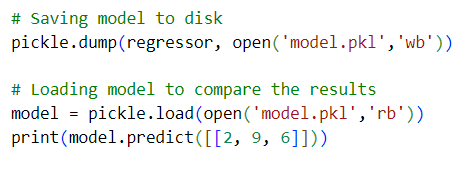
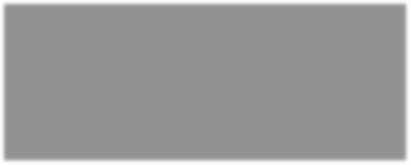
## Build a Model

The code below splits the dataset into training and testing sets, but since the dataset is very small, the model will be trained on all available data. It then imports the LinearRegression class from the Scikit-learn library and creates an instance of it. Finally, the model is fit to the training data using the fit() method.



## Save the Model

After that we save our model using pickle



# Turning Model into Web Application

Table 3.1: Application Folder File Directory

**app.py template/**

index.html

**static/css/**

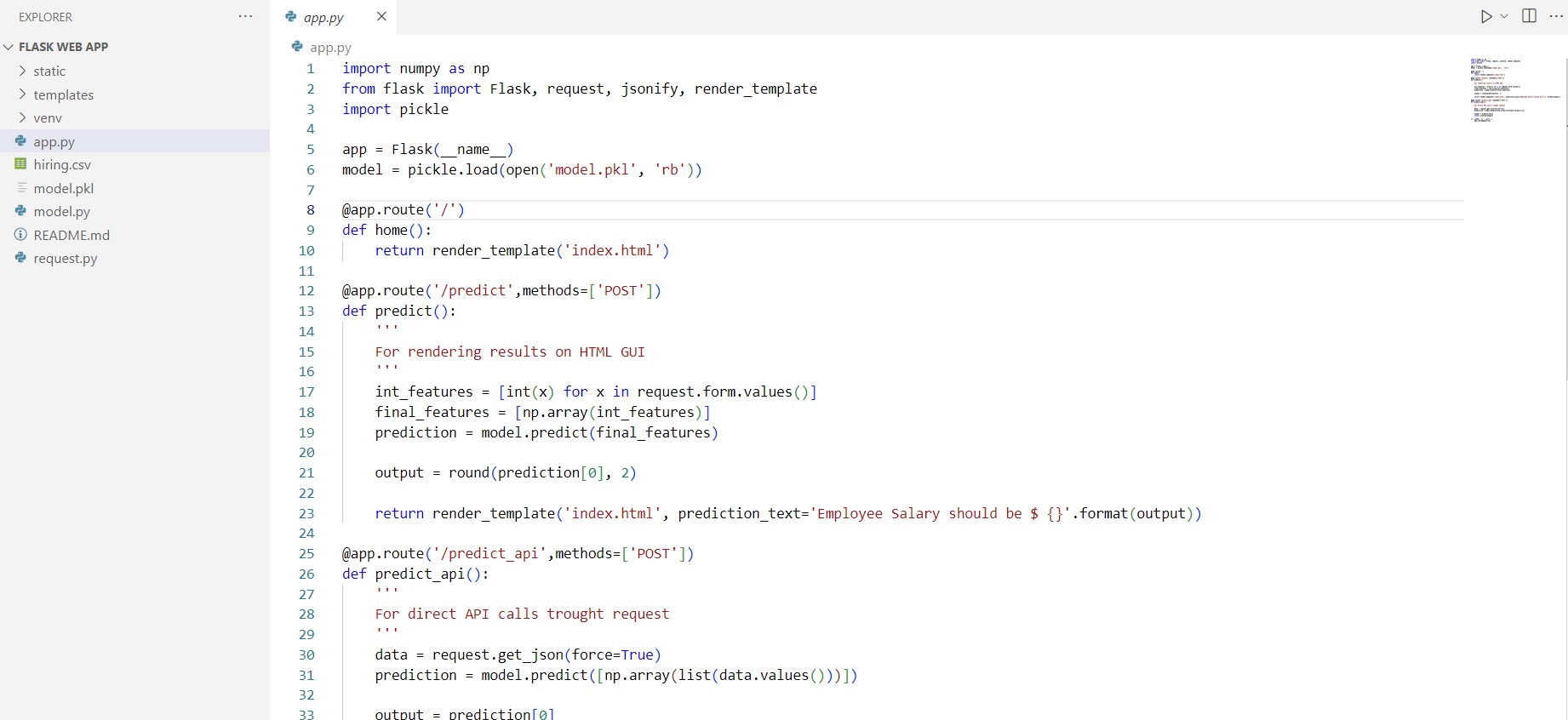
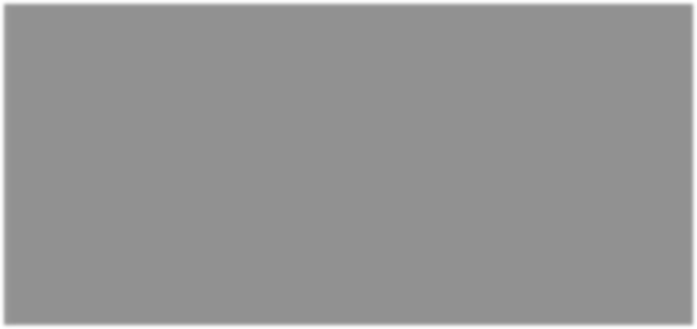
style.css **model.pkl hiring.csv**

**request.py**

## App.py

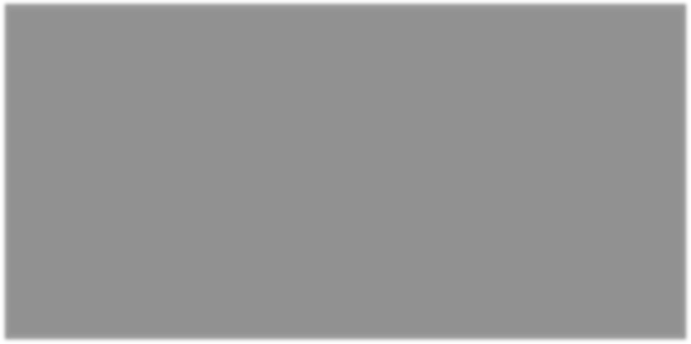
The app.py file contains the code that will be executed by the Python interpreter to run the Flask web application. The code creates a web application using Flask, a Python web application framework, and a machine learning model to predict employee salary.

The model is loaded from a saved file, and two routes are defined to handle HTTP requests. The '/predict' route accepts data submitted through an HTML form and returns a predicted salary value. The '/predict\_api' route accepts data submitted in JSON format and returns a salary prediction directly. The app.py file also includes a home route that renders an HTML template for the user to submit data. The output from the model prediction is rendered on the same page.



## index.html

The following are the contents of the index.html file that will render a text form where a user can enter his values.

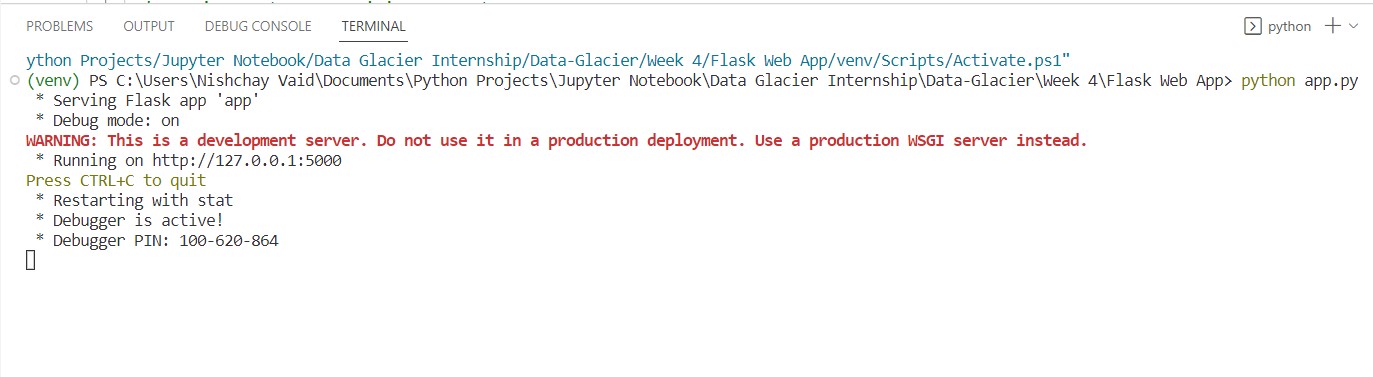
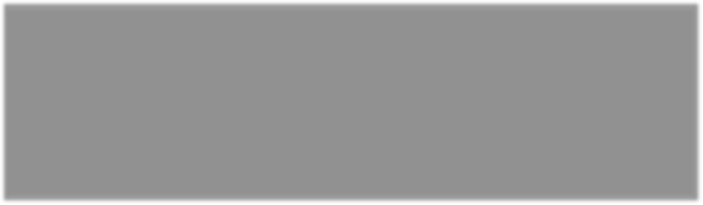


## Style.css

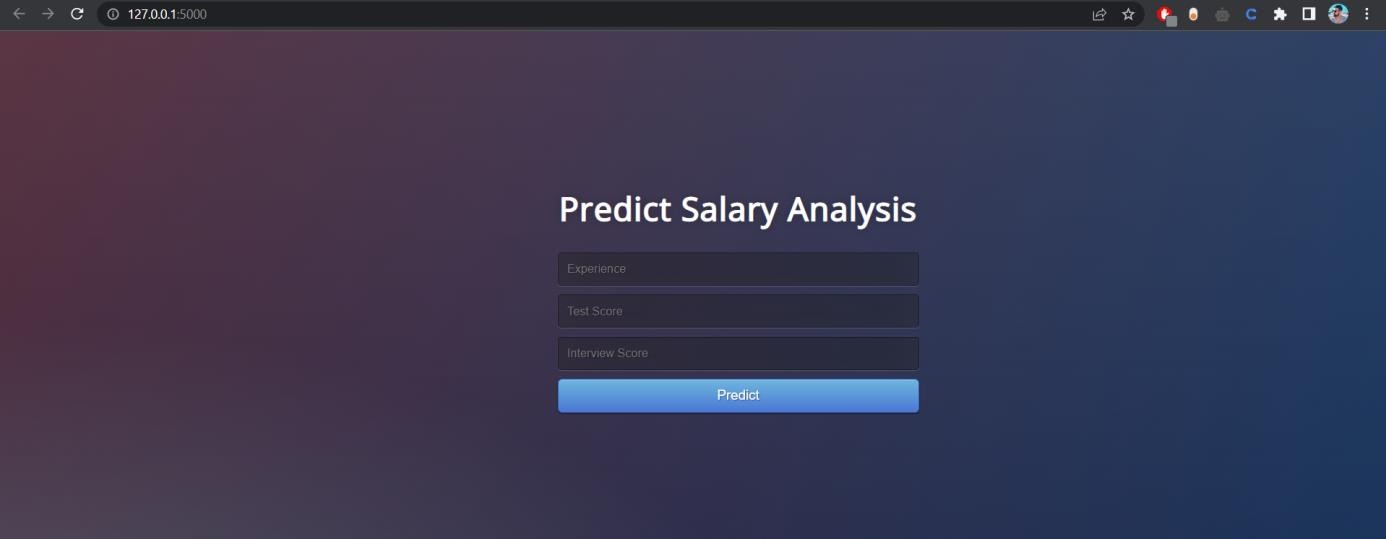
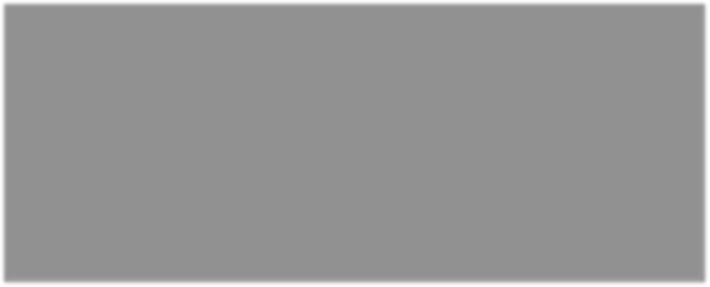
The index.html file includes a header section where the styles.css file is loaded. This file contains the styling information for the HTML document, and it should be saved in the 'static' sub-directory. Flask automatically looks for static files, such as CSS, in this directory.

## Running Procedure

Now, to start the Flask server we use the terminal and type in the command python app.py



After running the web application, we can open a web browser and enter the URL 'http://127.0.0.1:5000/'. This should display a basic website with a simple layout and design.



Now we enter input in the comments form and get the predicted employee salary

