Assignment: 1

Aim: Deploying ML model on Web with Flask.

Step 1: Install Flask

- To deploy model, flask package of python language is required. To install it write below command in command prompt.

```
PS E:\D2D studies\6th sem\ML\Codes> pip install flask
Collecting flask
  Obtaining dependency information for flask from https://files.pythonhosted.org/packages/93/a6/aa98bfe0eb9b8b1
5d36cdfd03c8ca86a03968a87f27ce224fb4f766acb23/flask-3.0.2-py3-none-any.whl.metadata
  Using cached flask-3.0.2-py3-none-any.whl.metadata (3.6 kB)
Requirement already satisfied: Werkzeug>=3.0.0 in c:\users\pratik\appdata\local\programs\python\python312\lib\s
ite-packages (from flask) (3.0.1)
Requirement already satisfied: Jinja2>=3.1.2 in c:\users\pratik\appdata\local\programs\python\python312\lib\sit
e-packages (from flask) (3.1.3)
Requirement already satisfied: itsdangerous>=2.1.2 in c:\users\pratik\appdata\local\programs\python\python312\l
ib\site-packages (from flask) (2.1.2)
Requirement already satisfied: click>=8.1.3 in c:\users\pratik\appdata\local\programs\python\python312\lib\site
-packages (from flask) (8.1.7)
Requirement already satisfied: blinker>=1.6.2 in c:\users\pratik\appdata\local\programs\python\python312\lib\si
te-packages (from flask) (1.7.0)
Requirement already satisfied: colorama in c:\users\pratik\appdata\local\programs\python\python312\lib\site-pac
kages (from click>=8.1.3->flask) (0.4.6)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\pratik\appdata\local\programs\python\python312\lib\s
ite-packages (from Jinja2>=3.1.2->flask) (2.1.5)
Using cached flask-3.0.2-py3-none-any.whl (101 kB)
Installing collected packages: flask
Successfully installed flask-3.0.2

[notice] A new release of pip is available: 23.2.1 -> 24.0
[notice] To update, run: python.exe -m pip install --upgrade pip
PS E:\D2D studies\6th sem\ML\Codes>
```

Make sure that above screen will appear showing that flask installed successfully.

Step 2: Create Flask App and Model Logic.

- Create a new directory for your Flask app and navigate to it. Inside the directory, create a file named app.py and add the following:

```
66  @app.route('/', methods=['GET', 'POST'])
67  def index():
68      if request.method == 'POST':
69          # Get form data
70          years_experience = int(request.form['years_experience'])
71          salary = int(request.form['salary'])
72
73          # Predict position based on years of experience
74          predicted_position, plot_data = predict_position(salary, years_experience)
75
76          return render_template('result.html', years_experience=years_experience, salary=salary,
77                                 predicted_position=predicted_position, plot_data=plot_data)
78      else:
79          return render_template('index.html')
80
81  if __name__ == '__main__':
82      app.run(debug=True)
```

- Also, we can add Model code in app.py file. So, I am using K-nearest Neighbors Model and add following code:

ml_project > app.py > ...

```python
from flask import Flask, render_template, request
import pandas as pd
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import io
import base64

app = Flask(__name__)

# Load the dataset
data = pd.read_csv('ml_project\employees.csv')

# Drop unnecessary columns
data = data[['EMPLOYEE_ID', 'SALARY', 'POSITION', 'YEARS_EXPERIENCE']]

# Drop rows with missing values
data.dropna(inplace=True)

# Splitting the dataset into features and target variable
X = data[['SALARY', 'YEARS_EXPERIENCE']]
y = data['POSITION']
```

```python
# Splitting the dataset into the training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Feature scaling
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Training the KNN model
k = 5  # Number of neighbors to consider
knn = KNeighborsClassifier(n_neighbors=k)
knn.fit(X_train, y_train)


# Function to predict position and generate plot
def predict_position(salary, years_experience):
    # Prepare the input data
    input_data = scaler.transform([[salary, years_experience]])

    # Predict position
    position = knn.predict(input_data)[0]
```

```python
    # Generate plot
    plt.scatter(data['SALARY'], data['YEARS_EXPERIENCE'], color='blue', label='Actual')
    plt.scatter(salary, years_experience, color='red', label='Predicted')
    plt.title('Actual vs Predicted Position')
    plt.xlabel('Salary')
    plt.ylabel('Years of Experience')
    plt.legend()

    # Convert plot to base64 encoded image
    buffer = io.BytesIO()
    plt.savefig(buffer, format='png')
    buffer.seek(0)
    plot_data = base64.b64encode(buffer.getvalue()).decode('utf-8')
    buffer.close()

    return position, plot_data
```

Step 3: Create template folder and add html file into it.

- Inside your Flask app directory, create a folder named templates. This is where you'll store your HTML templates. Create a file named index.html inside the templates folder and include the HTML structure:

ml_project > templates > <> index.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Employee Position Prediction</title>
</head>
<body>
    <h1>Predict Employee Position</h1>
    <form action="/" method="post">
        <label for="years_experience">Years of Experience:</label>
        <input type="text" id="years_experience" name="years_experience">
        <br>
        <label for="salary">Salary:</label>
        <input type="text" id="salary" name="salary">
        <br>
        <input type="submit" value="Submit">
    </form>
</body>
</html>
```

- Here, index.html page takes input from user and it passes to the flask, and based on user input KNN model predicts the position and that shows on result.html.

ml_project > templates > result.html

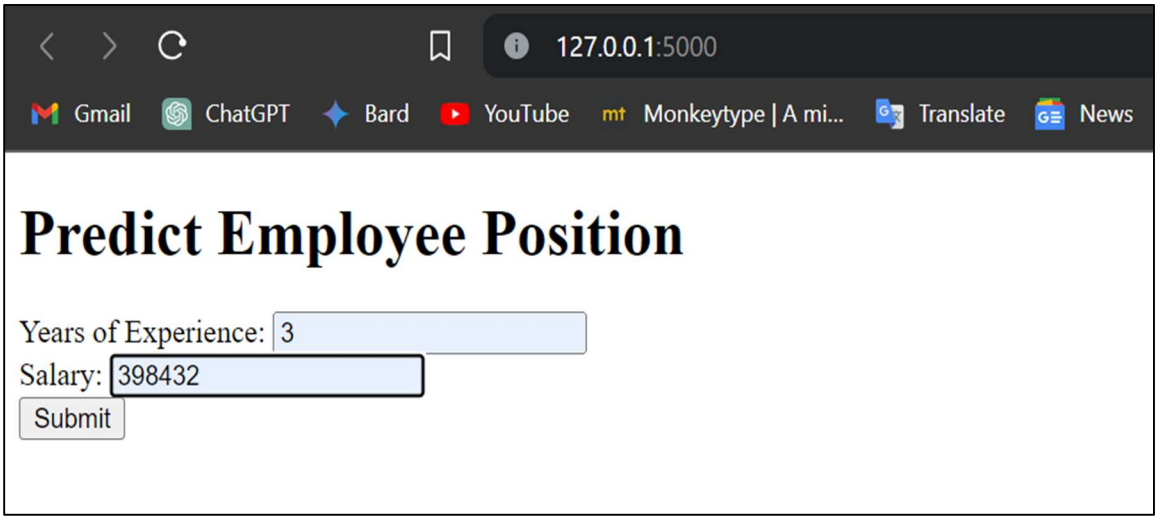
```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Salary Prediction Result</title>
</head>
<body>
    <h1>Salary Prediction Result</h1>
    <p>Years of Experience: {{ years_experience }}</p>
    <p>Salary: {{ salary }}</p>
    <p>Predicted Position: {{ predicted_position }}</p>
    <img src="{{ plot_data }}" alt="Graph">
</body>
</html>
```

Step 6: Run app.py

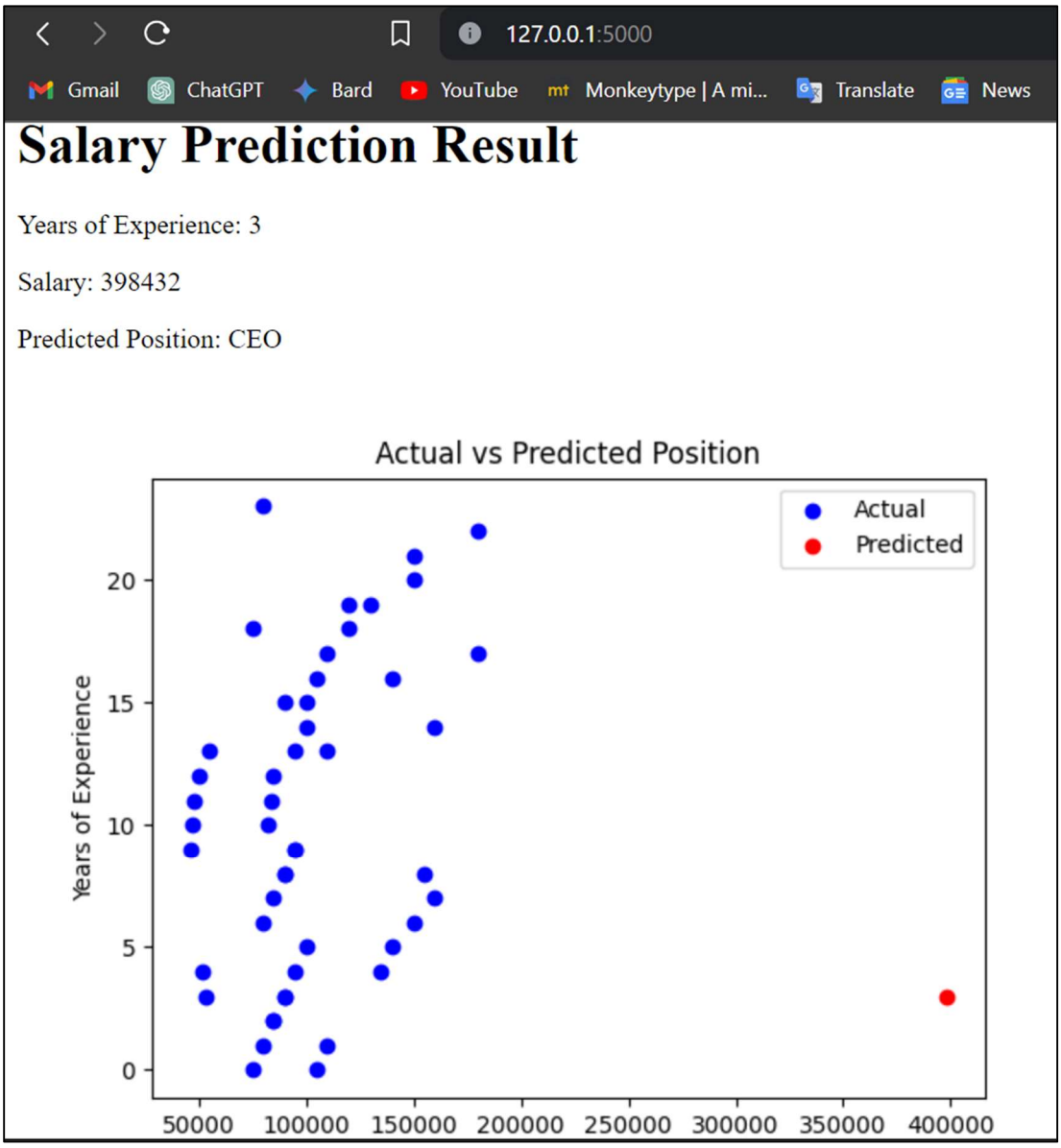
- This is final step. Type command **python –u app.py** in command prompt execute this python code, and also it generates the graph.
- Open web browser and go to http://127.0.0.1:5000/ to see your KNN Graph visualization.

OUTPUT:

Index.html



result.html


127.0.0.1:5000
Gmail
ChatGPT
Bard
YouTube
Monkeytype | A mi...
Translate
News
Salary Prediction Result
Years of Experience: 3
Salary: 398432
Predicted Position: CEO
Actual vs Predicted Position
Actual
Predicted
Years of Experience
0
5
10
15
20
50000
100000
150000
200000
250000
300000
350000
400000