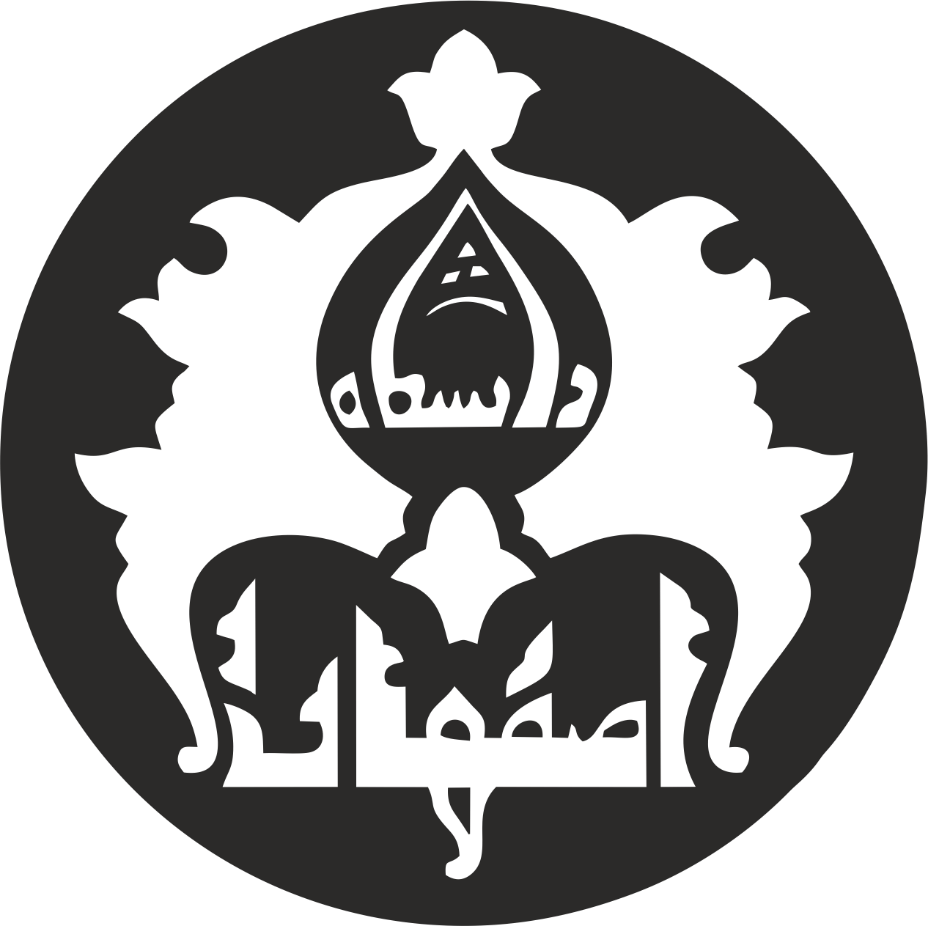
****

**گزارش پروژه اول قسمت اول**

**پروژه یافتن بهترین پرواز**

**درس: مبانی و کاربردهای هوش مصنوعی**

**استاد راهنما: دکتر حسین کارشناس نجف آبادی**

**اعضای گروه:**

**علی‌اکبر احراری- 4003613001**

**مهرآذین مزروق- 4003613055**

**پاییز 1402**

**گزارش کار الگوریتم و قسمت های مختلف کد**

**الگوریتم دایکسترا**

**dijkstra\_generated\_cost .1**

**def dijkstra\_generated\_cost(param):  
 distance = param['Distance']  
 price = param['Price']  
 time = param['FlyTime']  
 w2 = 0.3 # Weight for distance  
 w3 = 0.3 # Weight for price  
 w1 = 0.4 # Weight for time  
  
 cost = w1 \* time + w2 \* distance + w3 \* price  
 return cost**

**وظیفه این تابع، حساب کردن مقدار هزینه برای الگوریتم دایکستراست. این کار با استفاده از ترکیب خطی انتخابی، با سه پارامتر فاصله، قیمت و مدت زمان پرواز دو فرودگاه بدست می‌آید.**

**Args:**

**param (dict): A dictionary containing distance, price, and time values.**

**Returns:**

**float: The calculated cost.**

**2. add\_edge**

**def add\_edge(i):  
 if i == 1:  
 for i in range(df.size):  
 cost = a\_generate\_cost(df.iloc[i])  
 G.add\_edge(df.iloc[i, 1], df.iloc[i, 2], weight=cost)  
 else:  
 for i in range(12850):  
 cost = dijkstra\_generated\_cost(df.iloc[i])  
 G.add\_edge(df.iloc[i, 1], df.iloc[i, 2], weight=cost,  
 Distance=df.iloc[i, 13], FlyTime=df.iloc[i, 14], Price=df.iloc[i, 15])**

**این تابع، یال های گراف “G”را تشکیل می‌دهد. با انتخاب 1 به عنوان آرگومان، تابع یال ها را برای استفاده الگوریتم A\* و در صورت انتخاب 2 به عنوان آرگومان ورودی، تابع یال ها را برای استفاده الگوریتم Dijkstra به گراف اضافه می‌کند.**

**Args:**

**source (str): Name of the source airport.**

**target (str): Name of the target airport.**

**Returns:**

**list or None: A list representing the shortest path of airports, or None if no path is found.**

**3. dijkstra\_shortest\_path**

**def dijkstra\_shortest\_path(source, target):  
 try:  
 return nx.dijkstra\_path(G, source, target, weight='weight')  
 except nx.NetworkXNoPath:  
 return None**

**این تابع، کوتاه‌ترین مسیر بین یک مبدا و یک مقصد را با استفاده از الگوریتم دایکسترا محاسبه می‌کند.**

**Args:**

**source (str): Name of the source airport.**

**target (str): Name of the target airport.**

**Returns:**

**list or None: A list representing the shortest path of airports, or None if no path is found.**

**4. desired\_result\_string**

def desired\_result\_string(path):  
 flight\_number = 1  
 total\_time = 0  
 total\_price = 0  
 total\_distance = 0  
 result\_string = "" # Initialize an empty string to store the output  
 for u, v in zip(path, path[1:]):  
 edge\_data = G[u][v]  
 distance = round(edge\_data['Distance'])  
 price = round(edge\_data['Price'])  
 time = round(edge\_data['FlyTime'])  
 total\_time += time  
 total\_price += price  
 total\_distance += distance  
 if shortest\_path is not None:  
 result\_string += f'''  
 Flight #{flight\_number}:   
 From: {u}  
 To: {v}  
 Duration: {distance}km  
 Time: {time}h  
 Price: {price}$  
 ----------------------------'''  
 flight\_number += 1  
 else:  
 result\_string += "No path found."  
 result\_string += f'''  
 Total Duration: {total\_distance} km  
 Total Time: {total\_time}h  
 Total Price: {total\_price}$  
 '''  
 return result\_string # Return the result string

**این تابع، یک رشته به فرمت خروجی مورد نظر چاپ می‌کند.**

**Args:**

**path (list): List of airports representing the path.**

**Returns:**

**str: Formatted string containing flight details.**