

گزارش پروژه اول قسمت دوم

پروژه رگرسیون خطی چندمتغیره

درس: مبانی و کاربردهای هوش مصنوعی

استاد راهنما: دکتر حسین کارشناس نجف آبادی

اعضای گروه:

علی‌اکبر احراری- 4003613001

مهرآذین مزروق- 4003613055

پاییز 1402

فهرست

[گزارش کار الگوریتم و قسمت‌های مختلف کد 3](#_Toc150797730)

[df\_creator 3](#_Toc150797731)

[define\_and\_normalize\_xs 5](#_Toc150797732)

[gradiant\_descent 7](#_Toc150797733)

[generate\_errors 8](#_Toc150797734)

[generate\_file 9](#_Toc150797735)

[main codes 10](#_Toc150797736)

[نمونه خروجی 11](#_Toc150797737)

[کتابخانه‌های استفاده شده 12](#_Toc150797738)

[منابع 13](#_Toc150797739)

# گزارش کار الگوریتم و قسمت‌های مختلف کد

## df\_creator

def df\_creator():  
 file\_name = './Flight\_Price\_Dataset\_Q2.csv'  
 data\_frame = pd.read\_csv(file\_name)  
 encoded\_df = pd.DataFrame()  
  
 mapping = {'zero': 3,  
 'one': 2,  
 'two\_or\_more': 1}  
 encoded\_df['stops\_mapping'] = data\_frame['stops'].map(mapping)  
  
 mapping = {'Economy': 1,  
 'Business': 2}  
 encoded\_df['class\_mapping'] = data\_frame['class'].map(mapping)  
  
 mapping = {'Morning': 'morning\_departure',  
 'Early\_Morning': 'early\_morning\_departure',  
 'Evening': 'evening\_departure',  
 'Night': 'night\_departure',  
 'Afternoon': 'afternoon\_departure',  
 'Late\_Night': 'late\_night\_departure'}  
 data\_frame['departure\_mapping'] = data\_frame['departure\_time'].map(mapping)  
 departure\_dummies = pd.get\_dummies(data\_frame['departure\_mapping'])  
  
 mapping = {'Morning': 'morning\_arrival',  
 'Early\_Morning': 'early\_morning\_arrival',  
 'Evening': 'evening\_arrival',  
 'Night': 'night\_arrival',  
 'Afternoon': 'afternoon\_arrival',  
 'Late\_Night': 'late\_night\_arrival'}  
 data\_frame['arrival\_mapping'] = data\_frame['arrival\_time'].map(mapping)  
 arrival\_dummies = pd.get\_dummies(data\_frame['arrival\_mapping'])  
  
 encoded\_df = pd.concat([encoded\_df, departure\_dummies, arrival\_dummies, data\_frame['duration'],  
 data\_frame['days\_left'], data\_frame['price']], axis=1)  
 return encoded\_df

این تابع، ابتدا دیتافریمی از دیتاست سوال می‌سازد.

از این دیتاست، داده‌ها ستون stops را به اعداد ۱و۲و۳ مپ می‌کند.

ستون class را به اعداد ۱ و ۲ مپ می‌کند.

ستون‌های departure\_time و arrival\_time را نیز به روش One-Hot مپ می‌کند.

سپس این ستون‌های مپ شده، به‌علاوه‌ی ستون‌های duration و days\_left به یک دیتافریم جدید تبدیل می‌شوند و به عنوان دیتافریم اصلی پروژه معرفی می‌شوند.

## define\_and\_normalize\_xs

def define\_and\_normalize\_xs():  
 a1 = df['stops\_mapping']  
 a2 = df['class\_mapping']  
 a3 = df['duration']  
 a4 = df['days\_left']  
 a5 = df['afternoon\_departure']  
 a5 = a5.astype(int)  
 a6 = df['early\_morning\_departure']  
 a6 = a6.astype(int)  
 a7 = df['evening\_departure']  
 a7 = a7.astype(int)  
 a8 = df['late\_night\_departure']  
 a8 = a8.astype(int)  
 a9 = df['morning\_departure']  
 a9 = a9.astype(int)  
 a10 = df['night\_departure']  
 a10 = a10.astype(int)  
 a11 = df['afternoon\_arrival']  
 a11 = a11.astype(int)  
 a12 = df['early\_morning\_arrival']  
 a12 = a12.astype(int)  
 a13 = df['evening\_arrival']  
 a13 = a13.astype(int)  
 a14 = df['late\_night\_arrival']  
 a14 = a14.astype(int)  
 a15 = df['morning\_arrival']  
 a15 = a15.astype(int)  
 a16 = df['night\_arrival']  
 a16 = a16.astype(int)  
 # Normalize features  
 a1 = (a1 - a1.mean()) / a1.std()  
 a2 = (a2 - a2.mean()) / a2.std()  
 a3 = (a3 - a3.mean()) / a3.std()  
 a4 = (a4 - a4.mean()) / a4.std()  
 a5 = (a5 - a5.mean()) / a5.std()  
 a6 = (a6 - a6.mean()) / a6.std()  
 a7 = (a7 - a7.mean()) / a7.std()  
 a8 = (a8 - a8.mean()) / a8.std()  
 a9 = (a9 - a9.mean()) / a9.std()  
 a10 = (a10 - a10.mean()) / a10.std()  
 a11 = (a11 - a11.mean()) / a11.std()  
 a12 = (a12 - a12.mean()) / a12.std()  
 a13 = (a13 - a13.mean()) / a13.std()  
 a14 = (a14 - a14.mean()) / a14.std()  
 a15 = (a15 - a15.mean()) / a15.std()  
 a16 = (a16 - a16.mean()) / a16.std()  
 return np.c\_[a1, a2, a3, a4, a5, a6, a7, a8, a9, a10, a11, a12, a13, a14, a15, a16]

## gradiant\_descent

def gradiant\_descent(X, Y):  
 learning\_rate = 0.001  
 epochs = 2700  
 N = Y.size  
 coeff = np.random.rand(17)  
 past\_costs = []  
 PAST\_COEFF = [coeff]  
 for i in range(epochs):  
 prediction = np.dot(X, coeff)  
 error = prediction - Y  
 cost = 1 / (2 \* N) \* np.dot(error.T, error)  
 past\_costs.append(cost)  
 der = (1 / N) \* learning\_rate \* np.dot(X.T, error)  
 coeff = coeff - der  
 PAST\_COEFF.append(coeff)  
 return PAST\_COEFF, past\_costs

## generate\_errors

def generate\_errors():  
 # Predictions  
 predictions = np.dot(x\_test, coeffi)  
 # Mean Squared Error (MSE)  
 mse = np.mean((y\_test - predictions) \*\* 2)  
 # Root Mean Squared Error (RMSE)  
 rmse = np.sqrt(mse)  
 # R-squared (R²)  
 mean\_y = np.mean(y\_test)  
 ss\_total = np.sum((y\_test - mean\_y) \*\* 2)  
 ss\_residual = np.sum((y\_test - predictions) \*\* 2)  
 r\_squared = 1 - (ss\_residual / ss\_total)  
 # Mean Absolute Error (MAE)  
 mae = mean\_absolute\_error(y\_test, predictions)  
 return mse, rmse, mae, r\_squared

## generate\_file

def generate\_file():  
 st = "PRICE = "  
 for k in range(16):  
 st = st + f' ({coeffi[k]}) \* [{df.columns[k]}] +'  
  
 ans = st[:-1]  
 t = f'\nTraining Time: {round(end\_time - start\_time)}s'  
 MSE, RMSE, MAE, R\_SQUARED = generate\_errors()  
 errors = (f'\n\nLogs: '  
 f'\nMSE: {MSE}'  
 f'\nRMSE: {RMSE}'  
 f'\nMAE: {MAE}'  
 f'\nR2: {R\_SQUARED}')  
  
 file = open('[2]-UIAI4021-PR1-Q2.txt.txt', 'w')  
 file.write(ans)  
 file.write(t)  
 file.write(errors)  
 file.close()

این تابع، تابع خطی به‌دست آمده از الگوریتم را به صورت متن، وارد فایل می‌کند.

## main codes

df = df\_creator()  
# Define "y"  
y = df['price']  
x = np.c\_[define\_and\_normalize\_xs(), np.ones(270138)]  
x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.2, shuffle=True)  
start\_time = time.time()  
past\_coeff, past\_cost = gradiant\_descent(x\_train, y\_train)  
end\_time = time.time()  
coeffi = past\_coeff[-1]  
  
generate\_file()

در شروع برنامه، دیتافریم اصلی پروژه ساخته می‌شود. سپس همه‌ی x ها ساخته و یک ستون که فقط حاوی ۱ می‌باشد ساخته می‌شود؛ سپس الگوریتم کاهش شیب اجرا می‌شود و در نهایت فایل مورد نیاز ساخته می‌شود.

# نمونه خروجی

A screen shot of a computer

Description automatically generated

# کتابخانه‌های استفاده شده

time: برای به‌دست آوردن زمان اجرای الگوریتم کاهش شیب

pandas: برای استفاده از داده‌های سوال و تبدیل آن به دیتافریم و استفاده از داده‌های دیتافریم‌ها

numpy:

sklearn.metrics: برای به‌دست آوردن MAE

sklearn.model\_selection: جداسازی داده‌ها به دو قسمت train و test

# منابع

[Bing AI - Search](https://www.bing.com/search?form=NTPCHB&q=Bing+AI&showconv=1)

<https://www.khanacademy.org/math/multivariable-calculus/applications-of-multivariable-derivatives/optimizing-multivariable-functions/a/what-is-gradient-descent>

<https://youtu.be/lCOHri09YmM?si=GIPhdwRy-fxqrr_M>