機器學習 Machine Learning

期末報告

New York City Taxi Fare Prediction

第四組

組長：林學謙

組員：謝崇志、游鎮遠

壹、主題介紹

Kaggle 競賽名稱：New York City Taxi Fare Prediction

目的：以紐約市乘車資訊預測計程車車資

Features：

1. pickup_datetime　　乘客上車時間
2. pickup_longitude　　乘客上車經度
3. pickup_latitude　　乘客上車緯度
4. dropoff_longitude　乘客下車經度
5. dropoff_latitude　乘客下車緯度
6. passenger_count　　乘客人數

Target：

1. fare_amount　　車資

訓練資料總數：55,000,000

（圖一） New York City Taxi Fare Prediction 在 Kaggle 上的封面

貳、訓練資料

一、 導入與觀察資料：

先將訓練與測試資料導入，並轉成 pandas 的 dataframe 格式。以.describe()觀察訓練資料是否有不合理之處，其輸出結果如（圖二）。

```
# read data in pandas dataframe
df_train =  pd.read_csv('/kaggle/input/new-york-city-taxi-fare-prediction/train.csv', nrows = 1_800_000)

df_train.describe()
```

	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count
count	1.800000e+06	1.800000e+06	1.800000e+06	1.799987e+06	1.799987e+06	1.800000e+06
mean	1.134465e+01	-7.252749e+01	3.993207e+01	-7.252676e+01	3.993026e+01	1.684056e+00
std	9.849368e+00	1.308197e+01	8.157327e+00	1.290230e+01	1.068819e+01	1.315989e+00
min	-6.200000e+01	-3.377681e+03	-3.458665e+03	-3.383297e+03	-3.461541e+03	0.000000e+00
25%	6.000000e+00	-7.399208e+01	4.073493e+01	-7.399140e+01	4.073401e+01	1.000000e+00
50%	8.500000e+00	-7.398181e+01	4.075263e+01	-7.398016e+01	4.075314e+01	1.000000e+00
75%	1.250000e+01	-7.396713e+01	4.076710e+01	-7.396369e+01	4.076810e+01	2.000000e+00
max	1.273310e+03	2.856442e+03	2.621628e+03	3.414307e+03	3.345917e+03	2.080000e+02

（圖二） 訓練資料.describe()的輸出

經過觀察，可以發現到像是車資的最小值為負數，或乘客人數的最小值為 0 人、最大值高達兩百多人等，這些資料明顯為錯誤資料，所以將先以資料預處理將其刪除。

二、 資料預處理：

1. 處理乘客人數小於 0 人或大於 5 至 6 人以上的錯誤資料

經過觀察資料可得知，訓練資料中含有乘客人數為 0 人的錯誤資料，先予以刪除，且本組認為，一輛計程車的最大乘車人數應落在 5 至 6 人左右，所以也將乘車人數大於 5 至 6 人的資料刪除，如（圖三）。

```
df_train = df_train[df_train.passenger_count <= 5]
df_train = df_train[df_train.passenger_count > 0]
```

（圖三） 處理乘客人數小於 0 人或大於 5 至 6 人以上的錯誤資料

2. 處理車資小於 2.5 美元的錯誤資料

在六月二十九日，聆聽各組的課堂報告之前，本組在車資部分僅刪除車資為 0 美元的資料，但聽取其他組別報告後，發現紐約市計程車車資的基本上車票價為 2.5 美元，也就是 2.5 美元以下的資料即為錯誤資料，予以刪除，如（圖四）。

```
print('Old size: %d' % len(df_train))
df_train = df_train[df_train.fare_amount>2.5]
print('New size: %d' % len(df_train))
```

```
Old size: 1748325
New size: 1748325
```

（圖四） 處理車資小於 2.5 美元的錯誤資料

3. 刪除空值資料

檢測訓練資料中是否具有空值，若有，則刪除，如（圖五）。

```
print(df_train.isnull().sum())
print('------------------------------')
print('Old size: %d' % len(df_train))
df_train = df_train.dropna(how = 'any', axis = 'rows')
print('New size: %d' % len(df_train))
```

```
key                0
fare_amount        0
pickup_datetime    0
pickup_longitude   0
pickup_latitude    0
dropoff_longitude  0
dropoff_latitude   0
passenger_count    0
dtype: int64
------------------------------
Old size: 1748325
New size: 1748325
```

（圖五） 刪除空值資料

4. 處理座標範圍

在訓練前，本組認為需先確保訓練資料與測試資料的座標範圍一致，任何超過預測資料範圍的訓練資料都是多餘的，可以預先刪除。

I、 先獲取測試資料的座標範圍，也就是經緯度的最大值與最小值，如（圖六）與（圖七）。

```
# minimum and maximum longitude test set
min(df_test.pickup_longitude.min(), df_test.dropoff_longitude.min()), \
max(df_test.pickup_longitude.max(), df_test.dropoff_longitude.max())
```

```
(-74.263242, -72.986532)
```

（圖六） 獲取測試資料經度的最大值與最小值

```
# minimum and maximum latitude test
min(df_test.pickup_latitude.min(), df_test.dropoff_latitude.min()), \
max(df_test.pickup_latitude.max(), df_test.dropoff_latitude.max())
```

```
(40.568973, 41.709555)
```

（圖七） 獲取測試資料緯度的最大值與最小值

II、 創建截取座標範圍的函式，用以只保留訓練資料中，與測試資料座標範圍相同的資料，並定義座標範圍，如（圖八）。

```
def select_within_boundingbox(df, BB):
    return (df.pickup_longitude >= BB[0]) & (df.pickup_longitude <= BB[1]) & \
           (df.pickup_latitude >= BB[2]) & (df.pickup_latitude <= BB[3]) & \
           (df.dropoff_longitude >= BB[0]) & (df.dropoff_longitude <= BB[1]) & \
           (df.dropoff_latitude >= BB[2]) & (df.dropoff_latitude <= BB[3])

BB = (-74.5, -72.8, 40.5, 41.8)
```

（圖八） 創建截取座標範圍的函式，並定義座標範圍

III、 只保留訓練資料中，與測試資料座標範圍相同的資料，如（圖九）。

```
print('Old size: %d' % len(df_train))
df_train = df_train[select_within_boundingbox(df_train, BB)]
print('New size: %d' % len(df_train))
```

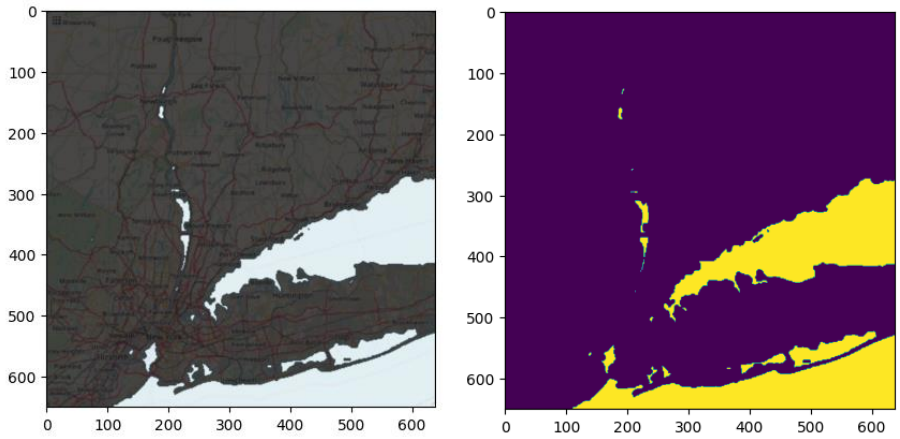
```
Old size: 1712472
New size: 1712472
```

（圖九） 只保留訓練資料中，與測試資料座標範圍相同的資料

5. 處理訓練資料中上下車座標在水域的錯誤資料

經過查閱 Kaggle 上許多先進的程式碼後，本組發現訓練資料中其實具有很多上下車座標在水域的錯誤資料，不合邏輯，必須予以刪除。

I、 載入紐約市在測試資料範圍的地圖，並建立藍顏色遮罩，依本組經驗，使用 nyc[:,:,0] > 0.678，如（圖十一），標記出水域範圍（圖十）。



（圖十） 載入紐約市在測試資料範圍的地圖 與 以遮罩標記出水域範圍

```
nyc_mask = nyc[:,:,0] > 0.678
```

（圖十一） 建立藍顏色遮罩

II、 創建將經緯度轉換成 xy 座標的函式，並以該函式觀察訓練資料中，有多少上下車座標在水域的錯誤資料，如（圖十二）。

```
# translate longitude/latitude coordinate into image xy coordinate
def lonlat_to_xy(longitude, latitude, dx, dy, BB):
    return (dx*(longitude - BB[0])/(BB[1]-BB[0])).astype('int'), \
           (dy - dy*(latitude - BB[2])/(BB[3]-BB[2])).astype('int')
```

```
pickup_x, pickup_y  = lonlat_to_xy(df_train.pickup_longitude, df_train.pickup_latitude,
                                   nyc_mask.shape[1], nyc_mask.shape[0], BB)
dropoff_x, dropoff_y = lonlat_to_xy(df_train.dropoff_longitude, df_train.dropoff_latitude,
                                   nyc_mask.shape[1], nyc_mask.shape[0], BB)
```

```
idx = (nyc_mask[pickup_y, pickup_x] & nyc_mask[dropoff_y, dropoff_x])
print("Number of trips in water: {}".format(np.sum(idx)))
```

```
Number of trips in water: 134
```

（圖十二） 創建將經緯度轉換成 xy 座標的函式，並以該函式觀察資料

III、 創建將上下車座標在水域的錯誤資料刪除的函式，並以該函式刪除資料，如（圖十三）。

```
def remove_datapoints_from_water(df):
    def lonlat_to_xy(longitude, latitude, dx, dy, BB):
        return (dx*(longitude - BB[0])/(BB[1]-BB[0])).astype('int'), \
               (dy - dy*(latitude - BB[2])/(BB[3]-BB[2])).astype('int')

    # define bounding box
    BB = (-74.5, -72.8, 40.5, 41.8)

    # read nyc mask and turn into boolean map with
    # land = True, water = False
    nyc_mask = plt.imread('/kaggle/input/masked-map/masked_map.png')[:,:,0] > 0.678

    # calculate for each lon,lat coordinate the xy coordinate in the mask map
    pickup_x, pickup_y = lonlat_to_xy(df.pickup_longitude, df.pickup_latitude,
                                      nyc_mask.shape[1], nyc_mask.shape[0], BB)
    dropoff_x, dropoff_y = lonlat_to_xy(df.dropoff_longitude, df.dropoff_latitude,
                                      nyc_mask.shape[1], nyc_mask.shape[0], BB)
    # calculate boolean index
    idx = nyc_mask[pickup_y, pickup_x] & nyc_mask[dropoff_y, dropoff_x]

    # return only datapoints on land
    return df[~idx]
```

```
print('Old size: %d' % len(df_train))
df_train = remove_datapoints_from_water(df_train)
print('New size: %d' % len(df_train))
```

```
Old size: 1718855

New size: 1718689
```

（圖十三） 創建將座標在水域的錯誤資料刪除的函式，並以該函式刪除資料

三、 資料標籤分析：

1. 新增 Features：

I、 新增小時、日、月、年、週的 Features。將原有 Feature 中給的時間表示，以 pandas 的.to_datetime 與.dt 轉換成個別的時間格式，共新增五個 Features。

II、 新增直線距離的 Feature。創建一個以 Haversine formula 將上下車座標轉換為直線距離的函式，並將直線距離作為新 Feature。

III、 新增到機場距離的 Features。經過查閱 Kaggle 上許多先進的程式碼後，本組發現可以將上下車座標與紐約市中三個機場的直線距離作為新 Features，於是創建一個轉換函式，共新增六個 Features。

2. 刪除 Features：

I、 刪除上下車座標 Features。本組認為，上下車的經緯度座標並不是一個數值，而是一個點位，對於預測車資並不是有用的訊息，所以予以刪除，而刪除後的測試結果為 RMSE 數值的確降低，誤差減少。

參、訓練模型

使用模型：XGBoost

Parameters：

Max depth = 6
estimator = 100

XGBoost（eXtreme Gradient Boosting）是一種在 Kaggle 競賽中常見的方法。XGBoost 結合了 Bagging 和 Boosting 的優點。

與 Gradient Boosting 類似，XGBoost 採用了逐步建構一個樹集合，每棵樹都會繼承前者的誤差，進行修正。越後面的樹，準確度會相對前面的樹提高。

此外 XGBoost 引入了特徵隨機採樣的技術，類似於隨機森林。在構建每棵樹時，會隨機選擇一個特徵子集，讓其多樣性更高。

總體而言，XGBoost 是一種強大的算法，結合了 Gradient Boosting 的優勢以及特徵隨機採樣等創新技術。這些改進使得 XGBoost 廣泛應用並在各種機器學習競賽中取得成功，成為數據科學家和實踐者的首選模型。

肆、結果

Submission score：

一、 2023/05/25

RMSE：**4.18999**

改進方法：

原本本組將直線距離為 0 的資料刪除，但後續測試過後發現，將距離為 0 的資料保留，預測誤差較小，可能是因為刪除了極短距離的資訊。另外，保留最大乘客人數為 5 人或 6 人的資料前處理，也讓 RMSE 有差別，最終結果呈現其實將乘客人數大於 5 人的資料刪除後，預測誤差較小。而至於訓練資料的筆數，本組最終採用了 1,800,000 的資料數量，為經過本組 tuning 過後最好的資料數量。

二、 2023/05/26

RMSE：**3.07073**

伍、未來展望

在實作過程中，因為在一開始的訓練問題打轉太久，導致後續沒什麼時間進行修正，仍有一些想法並沒有被實際執行，例如本組想用多個模型進行互相校正，或加上更多特徵的擷取，讓其可以辨認出更好的結果，亦或是使用曼哈頓距離取代直線距離，還有更多資料的前處理，比如其他組別提到的基本車資曾調漲，都是有望增加準確度的方法。

陸、感想

透過這次的期末競賽計劃，我們學到如何與同儕合作、討論，也更瞭解了整個機器學習的運作，期許以後若再遇上機器學習的問題，就知道如何進行資料預處理，也能懂得模型的選擇和結構設計。另外，六月二十九日的期末上台報告也讓本組成員學習到，在同一個機器學習題目上，更有著許多不同的方法可以解決問題，透過彼此分享想法學習到了非常多。