

Feature Modification

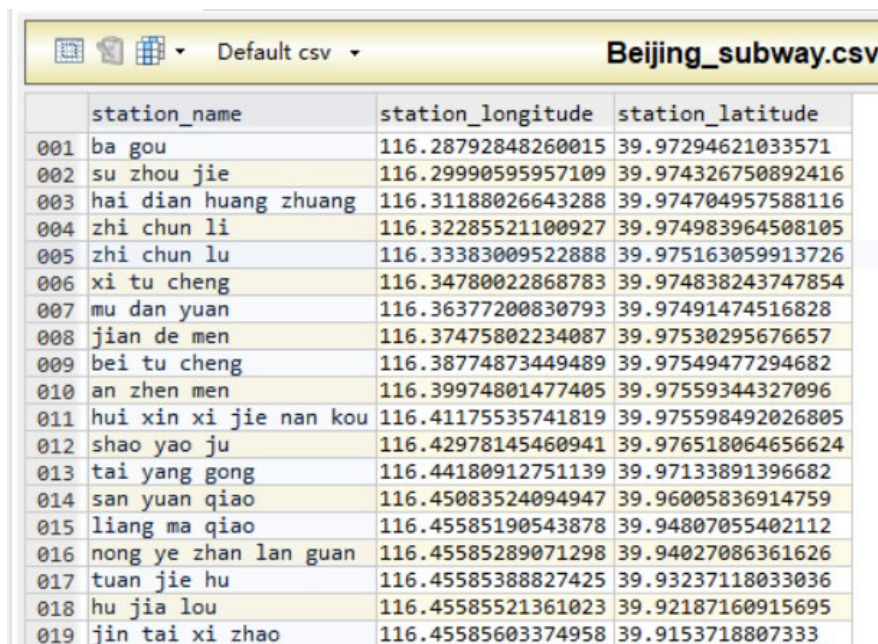
Key Features:

- Location: longitude and latitude of start locations and destinations.
- Time: request time including date, time, and day of a week, also peak hours, special time when some modes of public transportations are unavailable on the holidays or weekends.
- Plans: a list of feasible plans consist of transport mode (11 modes in total), estimated route distance in meters, ETA (estimated time of arrival) in seconds and estimated price in RMB shown by Baidu Maps to users.
- Profile: users' attributes related to preferences on different transport modes such as age, gender etc.
- Selection: (the first clicked transport mode) records of users' feedback of different recommendations shown in Baidu Maps.

Added Features:

- Related basic features
The Manhattan distance of start locations and destinations. The max, min, mean and std values of distance, price and ETA are calculated in every query session, and transport mode with the max/min of distance/price/ETA are recorded as well.
- Subway stations information
The name and locations of subway stations in Beijing are extracted by using ArcGis and converted into a csv file. This is a significant feature used to calculate the nearest distance from users' start point/destination to stations.

The related figures are shown as below:



	station_name	station_longitude	station_latitude
001	ba gou	116.28792848260015	39.97294621033571
002	su zhou jie	116.29990595957109	39.974326750892416
003	hai dian huang zhuang	116.31188026643288	39.974704957588116
004	zhi chun li	116.32285521100927	39.974983964508105
005	zhi chun lu	116.33383009522888	39.975163059913726
006	xi tu cheng	116.34780022868783	39.974838243747854
007	mu dan yuan	116.36377200830793	39.97491474516828
008	jian de men	116.37475802234087	39.97530295676657
009	bei tu cheng	116.38774873449489	39.97549477294682
010	an zhen men	116.39974801477405	39.97559344327096
011	hui xin xi jie nan kou	116.41175535741819	39.975598492026805
012	shao yao ju	116.42978145460941	39.976518064656624
013	tai yang gong	116.44180912751139	39.97133891396682
014	san yuan qiao	116.45083524094947	39.96005836914759
015	liang ma qiao	116.45585190543878	39.94807055402112
016	nong ye zhan lan guan	116.45585289071298	39.94027086361626
017	tuan jie hu	116.45585388827425	39.93237118033036
018	hu jia lou	116.45585521361023	39.92187160915695
019	jin tai xi zhao	116.45585603374958	39.9153718807333

The implemented code is shown as below:

```

def GCJ2WGS(location):
    # location format: locations[1] = "113.923745,22.530824"
    lon = float(location[0:location.find(",")])
    lat = float(location[location.find(",") + 1:len(location)])
    a = 6378245.0
    ee = 0.00669342162296594323
    PI = math.pi

    x = lon - 105.0
    y = lat - 35.0

    dLon = 300.0 + x + 2.0 * y + 0.1 * x * x + \
        0.1 * x * y + 0.1 * math.sqrt(abs(x))
    dLon += (20.0 * math.sin(6.0 * x * PI) + 20.0 *
        math.sin(2.0 * x * PI)) * 2.0 / 3.0
    dLon += (20.0 * math.sin(x * PI) + 40.0 *
        math.sin(x / 3.0 * PI)) * 2.0 / 3.0
    dLon += (150.0 * math.sin(x / 12.0 * PI) + 300.0 *
        math.sin(x / 30.0 * PI)) * 2.0 / 3.0

    dLat = -100.0 + 2.0 * x + 3.0 * y + 0.2 * y * \
        y + 0.1 * x * y + 0.2 * math.sqrt(abs(x))
    dLat += (20.0 * math.sin(6.0 * x * PI) + 20.0 *
        math.sin(2.0 * x * PI)) * 2.0 / 3.0
    dLat += (20.0 * math.sin(y * PI) + 40.0 *
        math.sin(y / 3.0 * PI)) * 2.0 / 3.0
    dLat += (160.0 * math.sin(y / 12.0 * PI) + 320 *
        math.sin(y * PI / 30.0)) * 2.0 / 3.0
    radLat = lat / 180.0 * PI
    magic = math.sin(radLat)
    magic = 1 - ee * magic * magic
    sqrtMagic = math.sqrt(magic)
    dLat = (dLat * 180.0) / ((a * (1 - ee)) / (magic * sqrtMagic) * PI)
    dLon = (dLon * 180.0) / (a / sqrtMagic * math.cos(radLat) * PI)
    wgsLon = lon - dLon
    wgsLat = lat - dLat
    return wgsLon, wgsLat

def parse_json(j):
    subway_pd = pd.DataFrame(
        columns=['station_name', 'station_longitude', 'station_latitude'])
    index = 0
    for line in j['l']:
        for station in line['st']:
            lg, la = GCJ2WGS(station['sl'])
            insert_list = [
                str(station['n']).strip(),
                float(lg),
                float(la)
            ]
            subway_pd.loc[index] = insert_list
            index += 1
    return subway_pd

```

- Weather information
Date, max and min temperatures, and weather mode.

max_temp	min_temp	weather	wind
18	4	0	12
8	-1	0	12
20	8	0	45
7	0	1	2
16	1	0	12
18	5	0	12
9	-1	1	12
25	10	0	12
8	1	2	34
14	2	1	12
19	4	0	34
25	10	0	12
14	2	0	12
25	10	0	12
18	7	1	12
17	7	3	12
7	-4	0	1
18	4	0	12
8	1	2	34
24	14	1	12
17	4	3	45

The code is shown as below:

```

1. def process(path):
2.     with open(path, 'r') as rf:
3.         weat = json.load(rf)
4.         path_, filename = os.path.split(path)
5.         output_path = filename.replace('.json', '.csv')
6.         weather_pd = parse_json(weat)
7.         weather_pd.to_csv(r'D:/UNIVERSITY/Junior_2nd/baidu/feature_data/'+output_path, index=False)
8.     print(output_path, 'done.')
9.
10.
11. def parse_json(j):
12.     weather_pd = pd.DataFrame(
13.         columns=['date', 'max_temp', 'min_temp', 'weather', 'wind'])
14.     index = 0
15.
16.     for k, v in j.items():
17.         insert_list = [k, int(v['max_temp']), int(
18.             v['min_temp']), v['weather'], int(v['wind'])]
19.         weather_pd.loc[index] = insert_list
20.         index += 1
21.
22.     return weather_pd

```

Feature Analysis

- In this project, the request time and the peak time of a day are not used as features which show no big influences on analysis results.
- There are 500k examples as train dataset and 94,358 examples as test dataset.