前言

1. 问题简介

在给定乘客上车时间，上下车经纬度以及乘客数量的情况下，探索纽约城出租车车费定价模型（包括通行费用）。

与出租车不同，网约车定价标准不固定，不唯一。在同样的分析框架下，假设我们有滴滴公司的公开数据集，我们将能探究"红包机制"对定价的影响。比如，探索红包在定价中的权重。再考虑到乘客发红包的行为往往与天气，交通堵塞等情况都有密切关系，所以我们还能探究不同情况下红包的大小，做定量比较。更进一步地，可以将探究出的网约车定价模型与出租车定价模型做比较，研究网约车定价是否合理。

2. 数据集简介

来源：https://www.kaggle.com/c/new-york-city-taxi-fare-prediction/data

```
import pandas as pd
path ='/Users/ness001/Downloads/train.csv'
train_data=pd.read_csv(path,nrows=10_000_000)
```

训练集总行数约为5500万行，为了方便探索，在此处选取前1000万行。

```
print(train_data.shape)
print(train_data.dtypes)
print(train_data.columns)
```

```
(10000000, 8)
key                 object
fare_amount        float64
pickup_datetime     object
pickup_longitude   float64
pickup_latitude    float64
dropoff_longitude  float64
dropoff_latitude   float64
passenger_count      int64
dtype: object
Index(['key', 'fare_amount', 'pickup_datetime', 'pickup_longitude',
       'pickup_latitude', 'dropoff_longitude', 'dropoff_latitude',
       'passenger_count'],
      dtype='object')
```

如上所示，训练集有8个特征值，由上到下分别代表：唯一标示值（由上车时间和唯一整数构成），支付金额，上车的时间，上车的经度，上车的纬度，下车的经度，下车的纬度，乘客人数。其中key和pickup_datatime为object类型变量，若要纳入模型，必须进行处理。

3. 实验环境

操作系统：macOS；软件：anaconda(py3.6) + jupyter notebook；使用包：pandas,sklearn,numpy

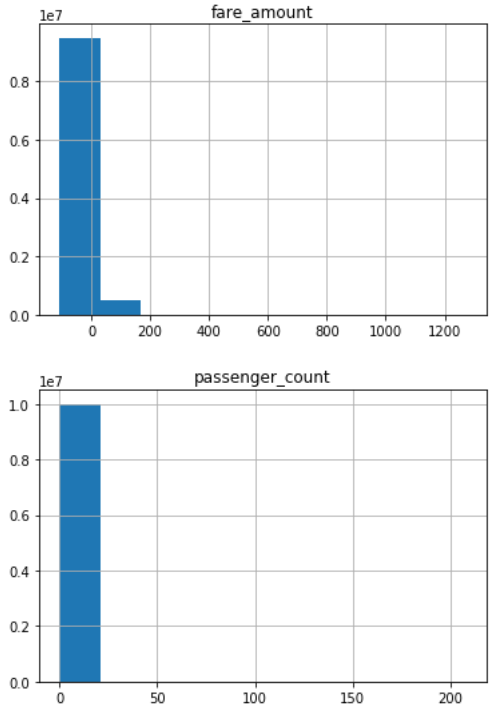
探索性数据分析

为了对数据获得直观印象，我们输出样本的基本统计信息，包括样本总数、均值、标准差、最大值、最小值和相关分位数，再做fare_amount和passenger_count的直方图，横坐标为变量的值的范围，纵坐标为频数。

```
print(train_data.describe())
print(train_data.hist('fare_amount'))
print(train_data.hist('passenger_count'))
```

```
        fare_amount  pickup_longitude  pickup_latitude  dropoff_longitude  \
count  1.000000e+07      1.000000e+07     1.000000e+07       9.999931e+06
mean   1.133854e+01     -7.250775e+01     3.991934e+01      -7.250897e+01
std    9.799930e+00      1.299421e+01     9.322539e+00       1.287532e+01
min   -1.077500e+02     -3.439245e+03    -3.492264e+03      -3.426601e+03
25%    6.000000e+00     -7.399207e+01     4.073491e+01      -7.399139e+01
50%    8.500000e+00     -7.398181e+01     4.075263e+01      -7.398016e+01
75%    1.250000e+01     -7.396710e+01     4.076712e+01      -7.396367e+01
max    1.273310e+03      3.457626e+03     3.344459e+03       3.457622e+03

       dropoff_latitude  passenger_count
count      9.999931e+06     1.000000e+07
mean       3.991913e+01     1.684793e+00
std        9.237280e+00     1.323423e+00
min       -3.488080e+03     0.000000e+00
25%        4.073403e+01     1.000000e+00
50%        4.075316e+01     1.000000e+00
75%        4.076810e+01     2.000000e+00
max        3.351403e+03     2.080000e+02
```



从统计表中，可以看出dropoff_longitude以及dropoff_latitude两个特征值的样本个数皆不足1000万，所以之后需要对缺失值做处理。支付金额最高逼近200美元。乘客人数大约在合理范围内，不排除有离群点，需要在之后进行排查。

建模前准备：数据修剪

1. 缺失值处理

检查各列有多少缺失值，选择处理方法。
常见的缺失值处理方法有
①删除法：这种方法简单易行，在对象有多个属性缺失值、被删除的含缺失值的对象与信息表中的数据量相比非常小的情况下是非常有效的。
②插值法：通常基于统计学原理，根据决策表中其余对象取值的分布情况来对一个空值进行填充，从而使信息表完备化。

```
print(train_data.isnull().sum())
```

```
key                   0
fare_amount           0
pickup_datetime       0
pickup_longitude      0
pickup_latitude       0
dropoff_longitude    69
dropoff_latitude     69
passenger_count       0
dtype: int64
```

与之前探索性数据分析中一致，下车的经纬度各缺了69个样本，缺失值量并不多，使用删除法直接删除这几行。检查行数以验证是否删除成功。

```
print('旧行数：%d' % len(train_data))
train_data = train_data.dropna(how = 'any', axis = 'rows')
print('新行数：%d' % len(train_data))
```

```
旧行数：10000000
新行数：9999931
```

2. 数值规范化

在数据集介绍部分，已经知道key和pickup_datetime这两个特征值的数据类型为object类型，且数值是标准的时间格式，所以可以将其直接转化为datetime类数据。两个变量表达的含义相同，我们只对key进行处理即可。在此基础上，才可以进一步将年、月、日、时等信息提取出来。

```
train_data['key'] = pd.to_datetime(train_data['key'])
```

```
data = [train_data]
for i in data:
    i['Year'] = i['key'].dt.year
    i['Month'] = i['key'].dt.month
    i['Date'] = i['key'].dt.day
    i['Day of Week'] = i['key'].dt.dayofweek
    i['Hour'] = i['key'].dt.hour
```

```
train_data.head(3)
```

	key	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude
0	2009-06-15 17:26:21.000000100	4.5	2009-06-15 17:26:21 UTC	-73.844311	40.721319	-73.841610	40.712278
1	2010-01-05 16:52:16.000000200	16.9	2010-01-05 16:52:16 UTC	-74.016048	40.711303	-73.979268	40.782004
2	2011-08-18 00:35:00.000000490	5.7	2011-08-18 00:35:00 UTC	-73.982738	40.761270	-73.991242	40.750562

3. 反常值处理

费用不可能是负数。检查删除后数据量。

```
train_data = train_data.drop(train_data[train_data['fare_amount']<0].index, axis=0)
train_data.shape
```

```
(9999511, 13)
```

经调查得，纽约出租车有SUV车型，所以最大乘客人数不可能超过六个。检查删除后数据量。

```
train_data = train_data.drop(train_data[train_data['passenger_count']>6].index, axis = 0)
train_data.shape
```

```
(9999494, 13)
```

纬度应在正负90之间，经度在正负180之间。检查删除后数据量。

```
train_data = train_data.drop(((train_data[train_data['pickup_latitude']<-90])|(train_data[train_data['pickup_latitude']>90])).index, axis=0)
train_data = train_data.drop(((train_data[train_data['pickup_longitude']<-180])|(train_data[train_data['pickup_longitude']>180])).index, axis=0)
train_data.shape
```

```
(9999193, 13)
```

4. 增加/减少特征

最初的几个特征值并未囊括车费定价中最重要的特征：路程长短，所以需要新增特征distance。
利用球面两点距离的半正矢公式，我们可以利用经纬度计算距离。

```
import numpy as np
def haversine_distance(lat1, long1, lat2, long2):
    data = [train_data]
    for i in data:
        R = 6371  #radius of earth in kilometers

        phi1 = np.radians(i[lat1])
        phi2 = np.radians(i[lat2])

        delta_phi = np.radians(i[lat2]-i[lat1])
        delta_lambda = np.radians(i[long2]-i[long1])

        #a = sin²((φB - φA)/2) + cos φA . cos φB . sin²((λB - λA)/2)
        a = np.sin(delta_phi / 2.0) ** 2 + np.cos(phi1) * np.cos(phi2) * np.sin(delta_lambda / 2.0) ** 2

        #c = 2 * atan2( √a, √(1−a) )
        c = 2 * np.arctan2(np.sqrt(a), np.sqrt(1-a))

        #d = R*c
        d = (R * c) #in kilometers
        i['distance'] = d
    return d
haversine_distance('pickup_latitude', 'pickup_longitude', 'dropoff_latitude', 'dropoff_longitude')
train_data.head(3)
```

	key	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count
0	2009-06-15 17:26:21.000000100	4.5	2009-06-15 17:26:21 UTC	-73.844311	40.721319	-73.841610	40.712278	1
1	2010-01-05 16:52:16.000000200	16.9	2010-01-05 16:52:16 UTC	-74.016048	40.711303	-73.979268	40.782004	1
2	2011-08-18 00:35:00.000000490	5.7	2011-08-18 00:35:00 UTC	-73.982738	40.761270	-73.991242	40.750562	2

5. 离群点处理

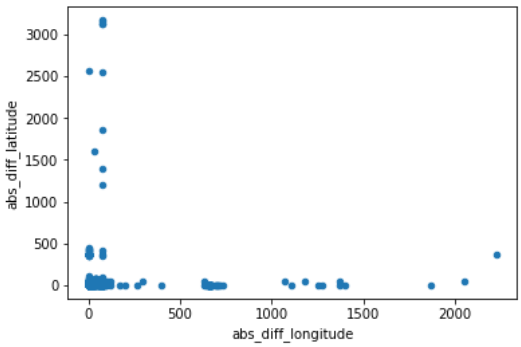
考虑到数据集是在纽约市内的乘车记录，经纬度之差应该不会太大。 我们新增两个特征：经度之差绝对值，纬度之差绝对值。

```
def add_travel_vector_features(df):
    df['abs_diff_longitude'] = (df.dropoff_longitude - df.pickup_longitude).abs()
    df['abs_diff_latitude'] = (df.dropoff_latitude - df.pickup_latitude).abs()
add_travel_vector_features(train_data)
```

做图观察分布，我们有理由相信那些离开“大部队”太远的少数分子是离群点。
按照图的分布，多次调整下面代码中“500”和“1500”这两个地方的数字，来删减数据。

```
plot = train_data.iloc[0:].plot.scatter('abs_diff_longitude', 'abs_diff_latitude')
print('旧数据量: %d' % len(train_data))
train_data = train_data[(train_data.abs_diff_longitude < 500) & (train_data.abs_diff_latitude < 1500)]
print('新数据量: %d' % len(train_data))
```

```
旧数据量: 9999193
新数据量: 9999083
```



检查一下数据量，发现删减并不多，相对合理，可以进入下一步。

初步建模

1. 确定目标变量和解释变量

为了方便之后优化模型(调参），我们对原来1000万行数据重排列后切片前1万条作为训练数据（相当于抽样）。

```
import numpy as np
train_data.reindex(np.random.permutation(train_data.index))
train_data=train_data[0:10000]
```

确定目标变量和解释变量。
解释变量为上下车经纬度，用来解释地点因素；乘客人数，距离，用来解释耗油因素；年月日时，解释时间在定价中的影响。

```
y=train_data.fare_amount
features=['pickup_longitude', 'pickup_latitude', 'dropoff_longitude','dropoff_latitude', 'passenger_count', 'distance', 'Year', 'Month', 'Date',
'Day of Week', 'Hour']
X=train_data[features]
```

2. 分离数据

为了方便做模型评估，分离出一部分数据作为验证集。

```
from sklearn.model_selection import train_test_split
train_X,val_X,train_y,val_y = train_test_split(X,y,random_state=1)
```

3. 训练模型

这是一个回归问题。在此处选用通用性良好的随机森林回归。

```
from sklearn.ensemble import RandomForestRegressor
betamodel=RandomForestRegressor(n_estimators=10,random_state=10)
betamodel.fit(train_X,train_y)
```

结果：

```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=None,
           oob_score=False, random_state=10, verbose=0, warm_start=False)
```

4. 模型评估

根据验证集，用模型预测出预测值。再使用rmse与r-square进行评估。

```
preds = betamodel.predict(val_X)
```

```
import math
from sklearn.metrics import mean_squared_error
print(math.sqrt(mean_squared_error(val_y,preds)))

from sklearn.metrics import r2_score
print(r2_score(val_y,preds))
```

```
4.7619626847341
0.7283932518011287
```

rmse较小，决定系数为0.72，不是特别理想。

优化模型

随机森林回归属于集成学习中的bagging框架，使用cart决策树作为弱学习器。
所以我们可以调整：1、bagging框架中的参数 2、cart决策树内的参数。

bagging框架中最需要关注的参数是：
1）n_estimators: 弱学习器的最大迭代次数。太小，容易欠拟合，太大，计算量会太大，并且n_estimators到一定的数量后，再增大n_estimators获得的模型提升会很小，所以一般选择一个适中的数值。默认是10。

cart决策树中的参数，在特征不多的情况下，只需调整：
1）min_samples_split: 这个值限制了子树继续划分的条件。如果某节点的样本数少于min_samples_split，则不会继续再尝试选择最优特征来进行划分。 默认是2，如果样本量数量级非常大，则推荐增大这个值。
2）min_samples_leaf: 这个值限制了叶子节点最少的样本数。如果某叶子节点数目小于样本数，则会和兄弟节点一起被剪枝。 默认是1,可以输入最少的样本数的整数，或者最少样本数占样本总数的百分比。如果样本量数量级非常大，则推荐增大这个值。

我们使用gridsearchCV寻找最优参数。

```
from sklearn.model_selection import GridSearchCV
param_test1 = {'n_estimators':range(10,180,10)}
gsearch1 = GridSearchCV(estimator = RandomForestRegressor(min_samples_split=2,
                                min_samples_leaf=1,random_state=10),
                        param_grid = param_test1,scoring='r2',cv=5)
gsearch1.fit(X,y)
print(gsearch1.best_params_, gsearch1.best_score_)
```

```
{'n_estimators': 130} 0.7582443295599112
```

```
from sklearn.model_selection import GridSearchCV
param_test2= {'min_samples_split':range(2,20,2),'min_samples_leaf':range(1,10,1)}
gsearch2 = GridSearchCV(estimator = RandomForestRegressor(n_estimators=130,random_state=10),
                        param_grid = param_test2,scoring='r2',cv=5)
gsearch2.fit(X,y)
print(gsearch2.best_params_, gsearch2.best_score_)
```

```
{'min_samples_leaf': 6, 'min_samples_split': 18} 0.7636925723800672
```

将上述挑选出的参数放入模型。效果达到预期，rmse下降约0.6，决定系数上升约0.07个百分点。

```
from sklearn.ensemble import RandomForestRegressor
newmodel=RandomForestRegressor(n_estimators=130,min_samples_leaf=6,min_samples_split=18,random_state=10)
newmodel.fit(train_X,train_y)
newpreds=newmodel.predict(val_X)
print(math.sqrt(mean_squared_error(val_y,newpreds)))

from sklearn.metrics import r2_score
print(r2_score(val_y,newpreds))
```

```
4.116210479299827
0.7970618354441205
```

小结与改进思路

受限于机能，为了快速调参，上述建模过程使用数据量较少。实际在导入100，000，000行数据的情况下，第一次训练结果的r2已经高达0.77，所以很显然，第一个改进思路是增加导入的数据。另外，我们使用的是bagging框架中的模型，因此第二个改进思路是改为使用boosting框架中的模型，比如lightGBM，通过调整boosting框架中的learning_rate等参数，效果可能会更好。