Kaggle Bike Sharing项目报告

标签： ML

项目定义

项目预览

项目地址

> https://www.kaggle.com/c/bike-sharing-demand

项目简介：

> Bike sharing systems are a means of renting bicycles where the process of obtaining membership, rental, and bike return is automated via a network of kiosk locations throughout a city. Using these systems, people are able rent a bike from a one location and return it to a different place on an as-needed basis. Currently, there are over 500 bike-sharing programs around the world.
> The data generated by these systems makes them attractive for researchers because the duration of travel, departure location, arrival location, and time elapsed is explicitly recorded. Bike sharing systems therefore function as a sensor network, which can be used for studying mobility in a city. In this competition, participants are asked to combine historical usage patterns with weather data in order to forecast bike rental demand in the Capital Bikeshare program in Washington, D.C.

城市租用自行车计划是在城市中部署若干个自助租车处。在这个由租车处组成的网络中使用者可自助租用、归还自行车。迄今为止，全世界已经有500多个自助自行车租用处。
这个租车系统产生的诸如租车时间、租借/交还位置，时间消费等数据引起了人们的关注。租车系统也借此成为了一个感知网络。本题目要求借助华盛顿的历史数据来预测租车系统的租借需求。

项目描述

本项目需要通过给予的历史数据（包括天气、时间、季节等特征）预测特定条件下的租车数目。通过选择决策树、SVM和随机森林算法，构建不同的模型。在特征过程后，使用训练数据集对模型行进行训练。最终使用训练好的模型对测试集进行预测。之后通过改变参数和使用交叉验证等方法提升模型精度。预测结果越靠近真实数据越好，在之后会介绍如何评判预测结果。

预测精度

项目提供了三个csv文件，分别是train.csv训练集，test.csv测试集，和sampleSubmission.csv上传文件格式.我们训练用到的是前两个文件。经过前面的分析，我认为这应该是一个监督学习中的回归问题。因为我们要输出一个个的离散值。我们通过数据采用一定的算法构建模型后，对test集中的数据进行预测。由于预测值的区间范围较大，因此采用网站给的评价准则，即：

Evaluation

Submissions are evaluated one the Root Mean Squared Logarithmic Error (RMSLE). The RMSLE is calculated as

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log(p_i+1)-\log(a_i+1))^2}$$

Where:

- n is the number of hours in the test set
- p_i is your predicted count
- a_i is the actual count
- $\log(x)$ is the natural logarithm

这个值越小越好。这个评价模型中同时考虑的预测值与实际值的差距，并没有考虑运行时间。为了满足RMSLE的条件，考虑在数据中对数据取对数之后在取e指数可以尽可能的满足此标准。

问题分析

数据概览

通过调用pd.read_csv()函数，读取train和test数据集。
通过

```
len(train)
len(test)
len(train.columns)
train.head()
```

得到训练集共10886条数据，测试集共6493条数据。共19个feature，分别为

```
[datetime, season, holiday, workingday, weather, temp, atemp, humidity,
windspeed]
```

以及输出的结果

```
[casual, registered, count]
```

数据样例如图

datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0	3	13	16
2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0	8	32	40
2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0	5	27	32
2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0	3	10	13
2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0	0	1	1
2011-01-01 05:00:00	1	0	0	2	9.84	12.88	75	6.0032	0	1	1
2011-01-01 06:00:00	1	0	0	1	9.02	13.635	80	0	2	0	2
2011-01-01 07:00:00	1	0	0	1	8.2	12.88	86	0	1	2	3
2011-01-01 08:00:00	1	0	0	1	9.84	14.395	75	0	1	7	8
2011-01-01 09:00:00	1	0	0	1	13.12	17.425	76	0	8	6	14
2011-01-01 10:00:00	1	0	0	1	15.58	19.695	76	16.9979	12	24	36

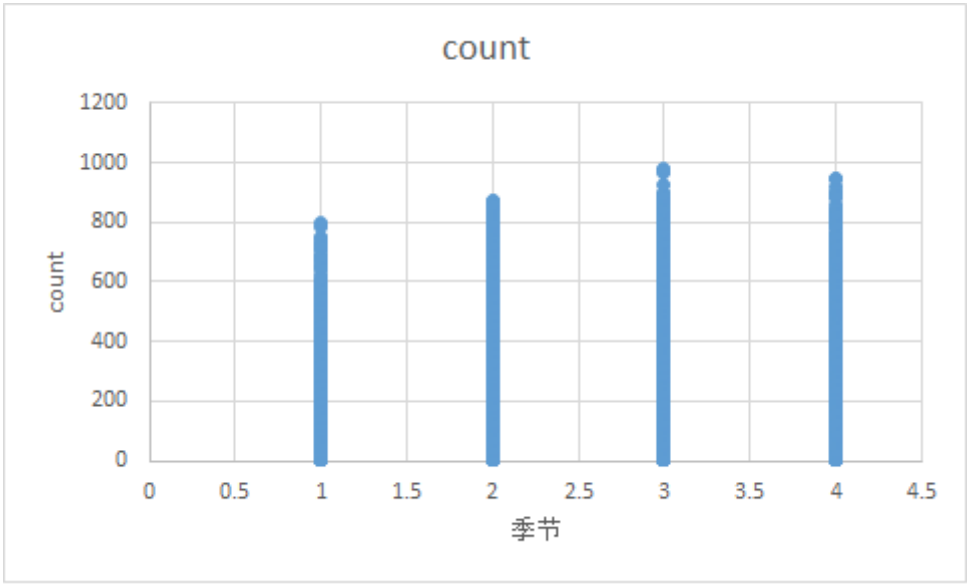
对特征进行分析，发现其中count为需要预测的值，casual和registered分别为登记和未登记的人数，加起来总和等于count，因此可以将其删除掉。
变量特征中

对特征进行分析，其中日期变量给出的是整体时间，考虑到租车与月份和日期和小时对预测是相对重要的，在接下来的数据处理中需要将其拆分成年/月/日/小时以及周几的形式。
季节特征(season)，在训练集是以category变量给出的，可以考虑将其转变成01变量。对租车的预测应该也是比较重要的。
天气变量(weather)同上。
训练集中给了temp/atemp两个变量，根据对atemp描述可知：

> atemp - “feels like” temperature in Celsius

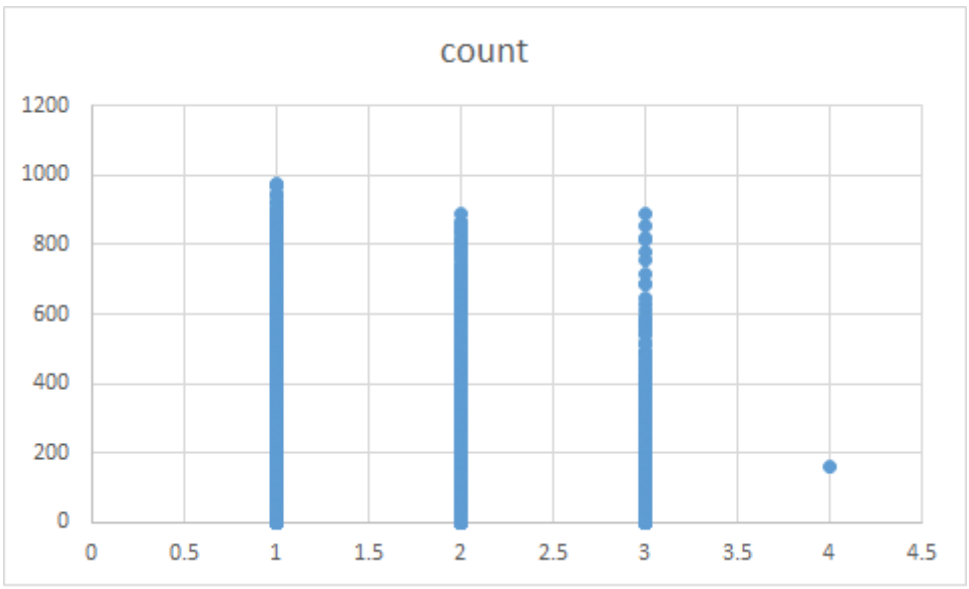
atemp是一个主观变量，和temp(即温度变量)有很高的相关性，考虑在数据处理中将其移除。

数据可视化
季节与count数关系柱状图



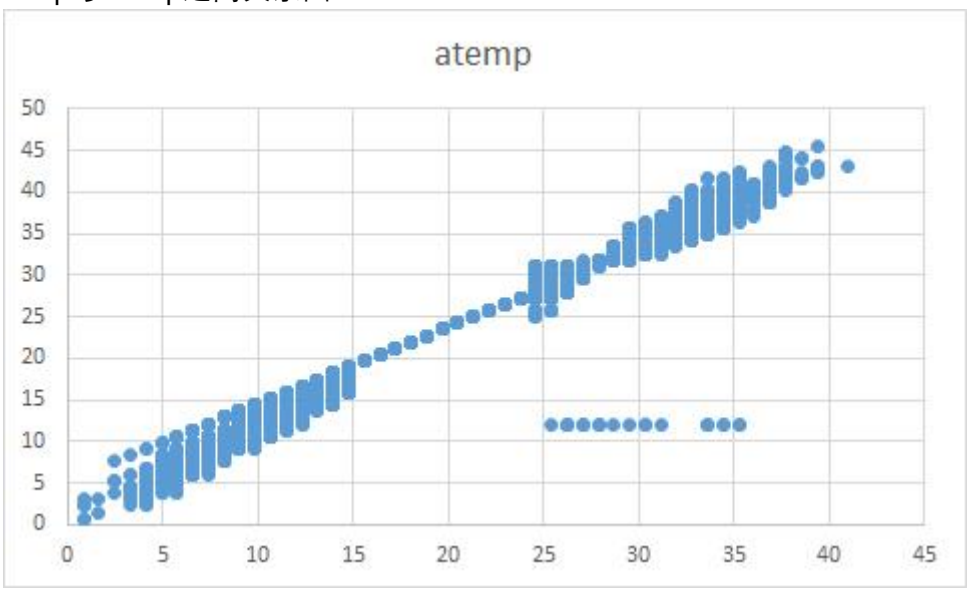
据此可以看出秋季是人们租车需求旺盛的季节。而春季是最低的季节。同时这是一个category变量，为了模型建立方便，考虑十栋dummies将其分解为四个01变量。

天气与count数关系柱状图



由此可以看出1类天气下租车人数最多，二三类接近，同为category变量，也将其分为三个01变量。

temp与atemp之间关系图



根据图像明显看出temp与atemp具有线性相关的关系，印证了上文得到的结论，因此为了避免维度灾

难，将atemp变量移除，只保留temp变量。

计算count的均值、中位数、四分位数。得到：

```
              count
count  10886.000000
mean     191.574132
std      181.144454
min        1.000000
25%       42.000000
50%      145.000000
75%      284.000000
max      977.000000
```

算法选取

考虑到本项目中是使用回归预测模型值，决定采用SVM、决策树，以及将决策树与聚成学习结合的随机森林算法。
SVM我之前知道的大多数用于分类，这里用于回归不知道效果如何。sklearn中的SVR负责回归，有C和epsilon两个参数，其中C为误差项的罚参数。epsilon代表epsilonSVR模型。在算法改进是使用交叉验证选取最优参数。
决策树算法比起线性回归要好得多，但也容易出现过拟合。而且当出现缺失值的情况时容易出现误判。主要参数为决策树最大层数max_depth。在算法改进是使用交叉验证选取最优参数。
随机森林算法结合了决策树和继承学习的优点，其计算简单容易实现。但是其最终集成的泛化性能会随着个体差异而增大，不过在本项目中足够使用了。主要参数为森林中树的数目，由于算法本身无需交叉验证，通过改变该参数调试最优结果。

基准点选择

考虑之后我认为本项目没有必要选取基准点，因为最坏情况是随机预测，这没有任何用处。

实施解决方案

1. 数据处理

首先按照上文所述，先对训练集和测试集进行Feature Engineering。
代码如下：

```
train['year'] = train['datetime'].str.extract("^(.{4})")
test['year'] = test['datetime'].str.extract("^(.{4})")

train['month'] = train['datetime'].str.extract("-(.{2})-")
test['month'] = test['datetime'].str.extract("-(.{2})-")

train['day'] = train['datetime'].str.extract("(.{2}) ")
test['day'] = test['datetime'].str.extract("(.{2}) ")

train['time'] = train['datetime'].str.extract(" (.{2})")
test['time'] = test['datetime'].str.extract(" (.{2})")

train[['year', 'month', 'day', 'time']] = train[['year', 'month', 'day',
'time']].astype(int)
test[['year', 'month', 'day', 'time']] = test[['year', 'month', 'day',
'time']].astype(int)

train['dayOfWeek'] = train.apply(lambda x: datetime.date(x['year'],
x['month'], x['day']).weekday(), axis=1)
test['dayOfWeek'] = test.apply(lambda x: datetime.date(x['year'], x['month'],
x['day']).weekday(), axis=1)

train = train.drop(labels=["datetime"], axis=1)
test = test.drop(labels=["datetime"], axis=1)
```

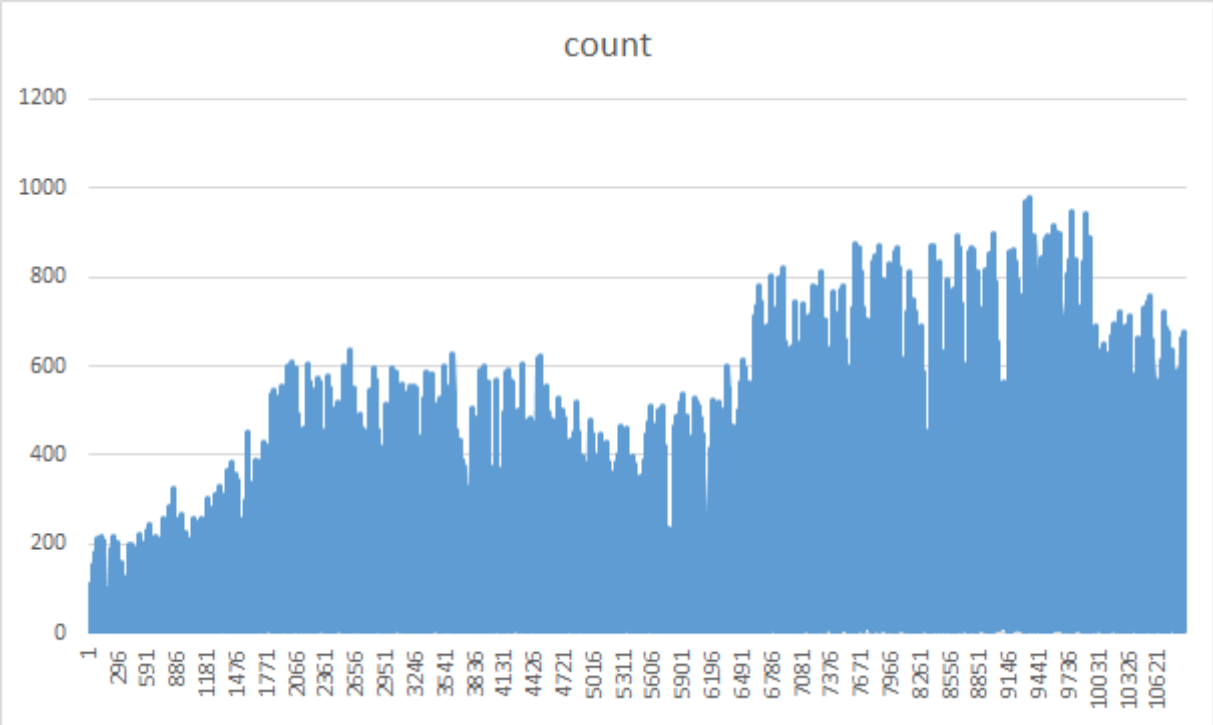
上述代码作用是处理datetime这一feature，将其转化为我们需要的年月日小时周几，将其转化为int类型后删除datetime。
对weather和season的处理即是对其使用get dummies函数使其生成多个0/1分类feature。

```
train['season'].value_counts()
train = train.join(pd.get_dummies(train.season, prefix='season'))
test = test.join(pd.get_dummies(test.season, prefix='season'))


train = train.drop(labels=["season"], axis=1)
test = test.drop(labels=["season"], axis=1)


train['weather'].value_counts()
train = train.join(pd.get_dummies(train.weather, prefix='weather'))
test = test.join(pd.get_dummies(test.weather, prefix='weather'))


train = train.drop(labels=["weather"], axis=1)
test = test.drop(labels=["weather"], axis=1)
```

上述步骤完成的是将datetime特征拆分成各个基本特征，将season，weather特征转变为01变量，最后去除datetime,weather,season,atemp特征。

数据处理完了之后，考虑到检验准确率的方式采用的是RMSLE，对train数据集取对数，进行预测之后再用指数函数变换回来。

根据数据分布图可知count数据范围变化不大，无须对count数据进行缩放变换。



2.建立模型并预测
为方便算法比较，生成一个函数读入回归模型，train数据，test数据来生成结果。

```
def bs_fit_and_save(clf, l_train, l_target, l_test, filename):

    clf.fit (l_train, l_target)

    predict_train = clf.predict(l_train)

    print('Variance score: %.2f' % clf.score(l_train, l_target))

    predict_test = clf.predict(l_test)
    predict_test = np.exp(predict_test)

    output = test_orig['datetime']
    output = pd.DataFrame(output)
    predict = pd.DataFrame(predict_test)
    output = output.join(predict)
    output.columns = ['datetime', 'count']
    output.to_csv(filename + ".csv", index=False)
    return clf
```

输出score来简单判断模型预测情况。不过此时的score只是显示对train数据的拟合程度。
这里decisionTree采用默认设置
randonForest参数设置n_estimators=100，
SVR设置C=1.0, epsilon=0.2

> decisionTree: Variance score: 1.00
> randomForest: 0.99
> SVR: 0.90

> 对分数分析，发现决策树很可能出现过拟合。随机森林和决策树分数都不错，SVR最低。再看一下kaggle给出的分数。
> 决策树分数
>
> | | | |
> |---|---|---|
> | - | danache | 0.56008 |
>
> 随机森林分数
>
> | | | |
> |---|---|---|
> | - | danache | 0.43617 |
>
> SVM分数：
>
> | | | |
> |---|---|---|
> | - | danache | 1.34572 |
>
> 由于在这里分数越低表现模型误差越小，因此可以看出来SVM最差，并不适合做此类分析，随机森林最好，决策树由于出现了过拟合情况表现也不佳。

模型改进

使用交叉验证，先对决策树使用使其max_depth参数范围为[1:10]，得道德结果score为

> Variance score: 0.96

kaggle验证分数为

-	danache	0.51382

相较于之前的0.56，提升了0.5左右，还是很明显的。
由于随机森林无需交叉验证，我用过改变n_estimators参数来修正模型，n_estimators从range(50,151,5)中选取。得到结果

> 50:0.43693
> 55:0.43583
> 60: 0.43599
> 70:0.43596
> 75:0.43655
> 80:0.43646
> 90:0.43621
> 90:0.43645
> 105:0.43528
> 110:0.43538
> 120:0.43540
> 130:0.43459

根据数据发现改变forest中tree的数量只能使预测误差微调，如果不改变数据的处理方式的话使用随机森林方法不能在本质上提升模型结果。由于tree的数目对结果影响不是现行的，因此我只能在上述参数中选

取最优的。
svm方法由于初始结果太差，放弃对其进行优化处理。
因此综上采用随机森林的方法可以较好的预测结果，最好的分数为0.43459。

结果

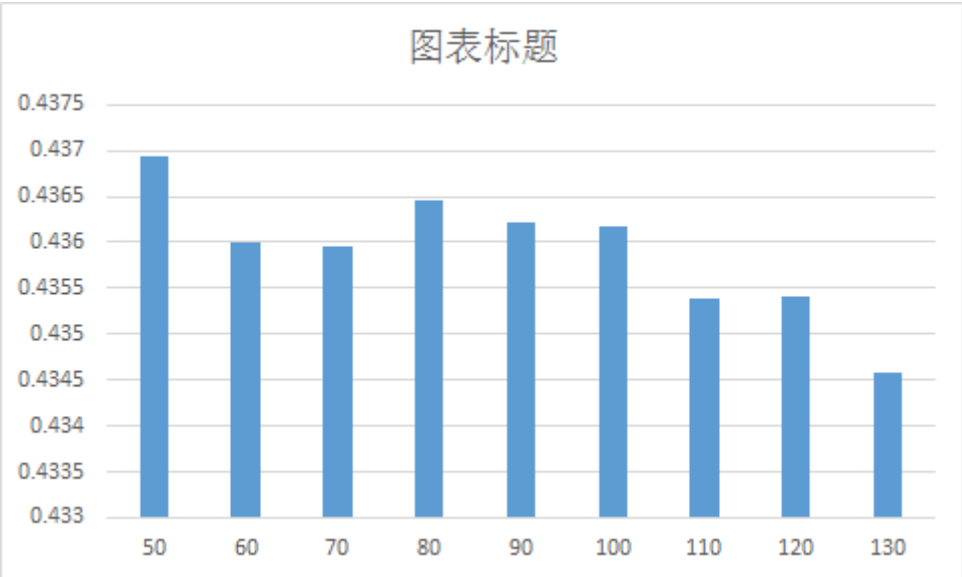
结果评估

通过上文的验证结果，最终选取的参数n_estimators=130的随机森林方法。其预测结果在kaggle上分数是最高的，为0.43459，排名698名。根据结果来说较好的满足的当初的预期。而其他两个模型不理想，我推测由于是模型算法原因，对于此类回归问题效果不太好。随机森林算法由于其本身的特性，使得其鲁棒性和抗数据扰动性能非常出色。因此采取此算法得到的结果我还是满意的。

总结

数据展示

选取随机森林中n_estimators参数作为自变量，验证数据拟合分数。



x轴为n_estimators的取值，y轴为kaggle网站数据拟合的分数。
发现随着n_estimators的增加，模型的拟合能力趋势为拟合分数越来越第，即性能越来越好，但是随着n_estimators数据拟合的时间越来越长，虽然项目中不考虑拟合时间，但是真实情况下时间成本还是需要

考虑的，当n_estimators为130时拟合时间已经相当长了。因此在增加变量数目对数据的提升不多。

反思

在完成本项目的过程中，我了解了以下几点：
①好的工作流程是必要的。一开始做的过程中，我只是参考之前的项目经验。然而边做边发现如果有一个好的流程框架对项目的提升是很大的。不论是项目结果还是完成时间
②data feature是至关重要的一环，甚至在很大程度上影响结果的精度。比如在本项目中给的datetime变量，由于其包含年月日小时等参数，如果不对其进行处理的话在模型预测中甚至是会有负收益。通过将其拆分并得出隐藏参数周数将其充分利用。而这个隐藏参数的挖掘需要对项目本身和实际情况有较好的理解。机器学习不仅是应用算法处理数据，还要加入自己对数据的理解和处理才能得到好的模型。
③模型选择也是比较重要的。通过上述讨论我们可以发现不同的模型对同一数据的预测结果差别是相当大的。在决策树模型中我发现当maxdepth为3时，对训练集的拟合精度可达到0.99，而预测集中，好多结果是相同的，而在真实情况中基本不可能发生这种情况。通过改变maxdepth可以在一定程度上改善这种问题，但是由于决策树本身的特性，过拟合还是无法避免的。
④对于结果还是比较满意的。训练结果很好，而且模型原因泛化能力较强。本项目目中的亮点是对season,weather特征的处理，一开始我自己的模型并没有对此进行处理，后来上了论坛之后才发现可以通过将其转为01变量提高精度。此外还有将其取对数进行分析最后在取指数的方法。根据题目不同的判断精度方式应该选取合适的处理方法。

提升

最终结果在kaggle达到了前20%，但是还是不够精确。这与自己经验、处理方式都有关系。在如何提高模型精度上我认为有以下几点：
①做好data feature，由于不知道更好的结果是如何做到，我猜测应该是对其进行了更多的特征转换等方法。如上文所述，好的data feature能很大程度上决定预测的精度。
②选取合适的模型。本项目中选取了决策树、SVM和随机森林。除此之外还有线性回归、神经网络等等方法没有尝试，或许使用卷积神经网络方法能对结果进行更好的预测？采用更好的模型也能提升预测的准确度。
③此外，就自行车租借而言，我们预测所需信息可能不仅仅是题目所给的，地理位置，附近有无居民区/商场等情况都应该考虑在内，限于题目的原因我们无法得知，在真实情况中我们应该考虑诸多变量仔细分析后才能得出精确的模型。