华东理工大学

模式识别大作业

题　　目　共享单车需求预测

学　　院　信息科学与工程

专　　业　控制科学与工程

姓　　名　杨建东

学　　号　Y30190764

指导教师　赵海涛

完成日期：　**2019** 年 **11** 月 **30** 日

目录

共享单车需求预测

一. 课题简介

1.1 课题背景

共享单车是近年来较为流行的出行方式，它的出现有效地缓解了城市道路交通的拥堵问题，为我国居民的出行带来了巨大的变革。然而共享单车随用随取、随停随还，这一特点导致它在发展过程中出现乱停乱放问题，对城市空间管理以及城市的美化造成了极大的影响。要想解决共享单车“乱”的问题，根本上是实现供需平衡，即共享单车的投放和用户的需求相适应、相匹配。

用户的需求是一个动态的过程，会随着各种因素的变化而变化，但通过对各因素的分析，用户的需求具有一定的可预测性，动态的调整共享单车区域投放数量、协调资源、智能调度、降低运营成本、提高用户体验、增强服务质量，对共享单车行业的可持续发展具有重大意义。

1.2 课题内容

本次作业来源于 kaggle 中 Bike Sharing Demand 比赛，要求结合历史使用模式和天气数据，预测华盛顿首都自行车租赁项目的租赁需求。数据集分为训练集 train.csv 和测试集 text.csv，其中训练集给出了影响自行车使用需求的 11 个影响因素以及最终的需求总数，测试集则给出了 9 个影响因素，要求预测最终的需求总数，预测的评判标准是均方根对数误差（RMSLE），$RMSLE=\sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log(p_i+1)-\log(a_i+1))^2}$。下面分别对 11 个影响因素和需求总数进行阐述：

表 1 共享单车数据信息

字段名称	字段说明
datatime	日期时间
season	季节
holiday	是否为节假日
workingday	是否为工作日
weather	天气状况
temp	实际温度（摄氏度）
atemp	体感温度（摄氏度）

humidity	空气湿度
windspeed	风速
casual	未注册用户租借数量
registered	注册用户租借数量
count	总租借数量

其中有几个字段的数据形式需要额外介绍一下：

datetime （即数据的日期时间）：年/月/日/时间，例：2019/1/1/12:35；

season （即四季）：1=春季，2=夏季，3=秋季，4=冬季；

holiday （是否为节假日）：0=否，1=是；

workingday （是否为工作日）：0=否，1=是；

weather （天气状况）：天气状况分为四个等级，其中 1=良好（晴天、多云等），2=普通（阴天薄雾等），3=稍差（小雪、小雨等），4=极差（大雨、冰雹等）。

二. 课题实施

首先导入训练数据集和测试数据集，判断数据集是否存在数据缺失现象，若存在数据缺失现象采用随机森林进行补全；接着对数据进行可视化分析，查找异常数据，并对不同的特征分别进行单变量分析、双变量分析以及部分的多变量分析，得到整体的相关分析关联度；再根据关联程度进行特征选择，分别采用 LinearRegression、RandomForestRegressor 以及 Lasso 回归模型对训练数据集进行训练，选出均方根对数误差最小的模型，建立回归模型对测试集数据做出预测。

2.1 数据导入

首先导入整个程序所用到的库函数包

```
# coding:utf-8
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.linear_model import LinearRegression,Ridge,Lasso
import warnings
pd.options.mode.chained_assignment = None
warnings.filterwarnings("ignore",category=DeprecationWarning)
```

```
pd.options.mode.chained_assignment = None
warnings.filterwarnings("ignore", category=DeprecationWarning)
```

接着分别导入训练数据集 train.csv 和测试数据集 test.csv，查看数据的具体类型和格式

```
DataTrain = pd.read_csv("../MS/train.csv")
DataTest = pd.read_csv("../MS/test.csv")
DataTrain.head()
DataTest.head()
DataTrain.info()
DataTest.info()
```

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	3	13	16
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	8	32	40
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	5	27	32
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0	3	10	13
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0	0	1	1

图 1 train.csv 部分数据集

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed
0	2011-01-20 00:00:00	1	0	1	1	10.66	11.365	56	26.0027
1	2011-01-20 01:00:00	1	0	1	1	10.66	13.635	56	0.0000
2	2011-01-20 02:00:00	1	0	1	1	10.66	13.635	56	0.0000
3	2011-01-20 03:00:00	1	0	1	1	10.66	12.880	56	11.0014
4	2011-01-20 04:00:00	1	0	1	1	10.66	12.880	56	11.0014

图 2 test.csv 部分数据集

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10886 entries, 0 to 10885
Data columns (total 12 columns):
datetime      10886 non-null object
season        10886 non-null int64
holiday       10886 non-null int64
workingday    10886 non-null int64
weather       10886 non-null int64
temp          10886 non-null float64
atemp         10886 non-null float64
humidity      10886 non-null int64
windspeed     10886 non-null float64
casual        10886 non-null int64
registered    10886 non-null int64
count         10886 non-null int64
dtypes: float64(3), int64(8), object(1)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6493 entries, 0 to 6492
Data columns (total 9 columns):
datetime      6493 non-null object
season        6493 non-null int64
holiday       6493 non-null int64
workingday    6493 non-null int64
weather       6493 non-null int64
temp          6493 non-null float64
atemp         6493 non-null float64
humidity      6493 non-null int64
windspeed     6493 non-null float64
dtypes: float64(3), int64(5), object(1)
```

图 3 数据集数据类型及数量

接着判断导入的两个数据集是否存在缺失值

```
DataTrain.isnull().sum()
DataTest.isnull().sum()
```

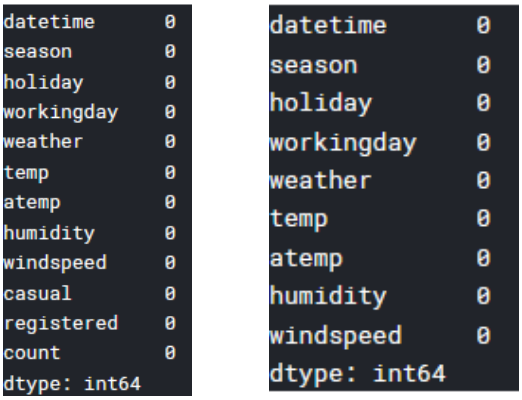


图 4 数据集变量缺失值统计

从上图中可以得到以下信息：训练数据集共 10886 个数据，每个数据包含 12 维特征，测试数据集共 6493 个数据，每个数据包含 9 维特征，两个数据集均不存在缺失值。

2.2 数据处理

2.2.1 数据切割

从图 1 和图 2 可以看出，数据集的 datetime 特征由年、月、日、时组成，实际上这一个特征包含了 4 个特征，考虑到用户租车与月份、日期以及小时段有密切关系，因此在数据处理阶段需要将其进行切割，加强预测的准确性。

```
# 训练数据集
DataTrain['datetime'] = pd.to_datetime(DataTrain['datetime'])
DataTrain['year'] = DataTrain['datetime'].dt.year
DataTrain['month'] = DataTrain['datetime'].dt.month
DataTrain['day'] = DataTrain['datetime'].dt.day
DataTrain['hour'] = DataTrain['datetime'].dt.hour
# 预测数据集
DataTest['datetime'] = pd.to_datetime(DataTest['datetime'])
DataTest['year'] = DataTest['datetime'].dt.year
DataTest['month'] = DataTest['datetime'].dt.month
DataTest['day'] = DataTest['datetime'].dt.day
DataTest['hour'] = DataTest['datetime'].dt.hour
```

2.2.2 异常值处理

数据集提供了诸多特征，这些特征值的部分数据可能存在异常值，以训练数据集为例，训练数据集包含 12 维特征，其中 season、holiday、workingday、weather 均已量化为离散的数字量，是否存在异常可以轻松判断，而 temp、atemp、humidity、windspeed 是连续变量，且根据我们日常体验，这些变量通常不会发生突变，这也为发现并处理异常值带来了可能，下面将这些变量的分布情况可视化。

```
fig1,(ax1, ax2) = plt.subplots(nrows=2, ncols=1)
fig1.set_size_inches(10, 5)
```

```
sns.regplot(x="index", y="temp", data=DataTrain, ax=ax1)
sns.regplot(x="index", y="atemp", data=DataTrain, ax=ax2)
ax1.set(ylabel="temp", title="temp scatter diagram")
ax2.set(ylabel="atemp", title="atemp scatter diagram")
fig2 = plt.figure()
fig2.set_size_inches(10, 5)
ax3 = fig2.add_subplot(1, 1, 1)
sns.regplot(x="index", y="humidity", data=DataTrain, ax=ax3)
ax3.set(ylabel="humidity", title="humidity scatter diagram")
fig3 = plt.figure()
fig3.set_size_inches(10, 5)
ax4 = fig3.add_subplot(1, 1, 1)
sns.regplot(x="index", y="windspeed", data=DataTrain, ax=ax4)
ax4.set(ylabel="windspeed", title="windspeed scatter diagram")
```

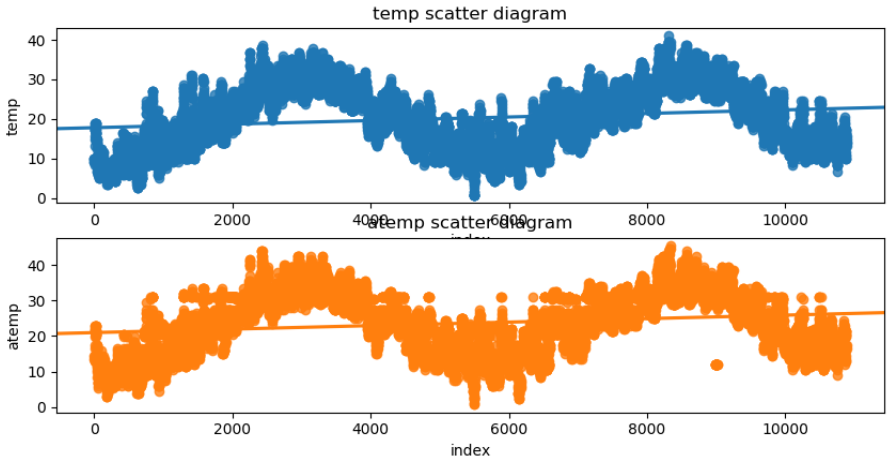
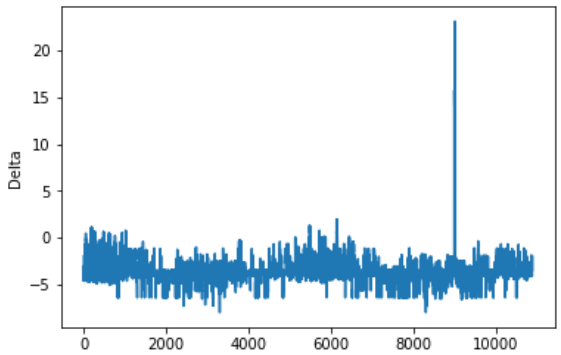


图 5 温度与体感温度分布图

从图中可以看出温度和体感温度的分布具有很高的相关性（相关性在 2.3 中讨论），但也存在部分离群点，进一步观察两者之间的区别：



	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	cour
8991	2012-08-17 00:00:00	3	0	1	1	27.88	12.12	57	11.0014	21	67	88
8992	2012-08-17 01:00:00	3	0	1	1	27.06	12.12	65	7.0015	16	38	54
8993	2012-08-17 02:00:00	3	0	1	1	27.06	12.12	61	8.9981	4	15	19
8994	2012-08-17 03:00:00	3	0	1	1	26.24	12.12	65	7.0015	0	6	6
8995	2012-08-17 04:00:00	3	0	1	1	26.24	12.12	73	11.0014	0	9	9

图 6 温度与体感温度差值

从上图可以明显得出离群点即为异常值，这里直接去除相应数据。

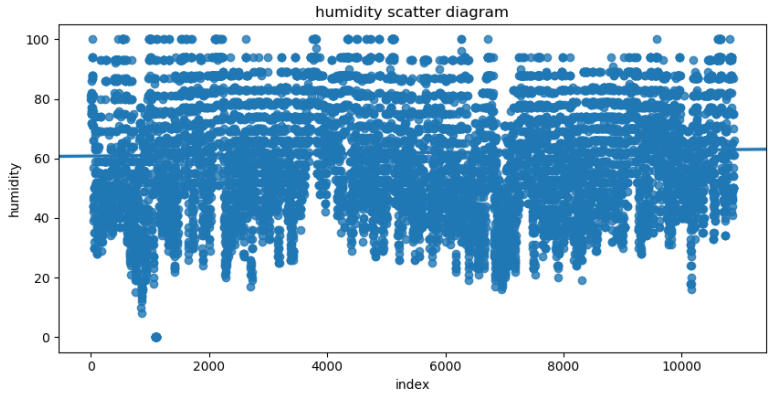


图 7 湿度分布图

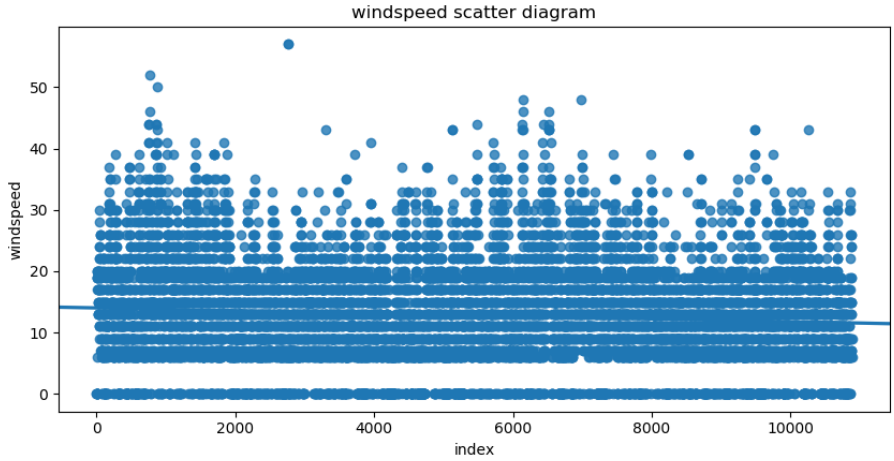


图 8 风速分布图

湿度和风速的分布不存在规律性，大部分数据散点较为密集，但也可以观察到风速存在大量的 0 点，这些散点偏离群体值，考虑到检测风速传感器的精度问题，猜测是由于风速较小，传感器精度不能检测出实际风速。

采用随机森林对这部分数据插值：

```
# 采用随机森林为异常风速 wind speed 赋值
dataWind0 = DataTrain[DataTrain["windspeed"] == 0]
dataWindNot0 = DataTrain[DataTrain["windspeed"] != 0]
rfModel_wind = RandomForestRegressor()
windColumns = ["season", "weather", "humidity", "month", "temp", "year", "atemp"]
rfModel_wind.fit(dataWindNot0[windColumns], dataWindNot0["windspeed"])

wind0Values = rfModel_wind.predict(X=dataWind0[windColumns])
dataWind0["windspeed"] = wind0Values
DataTrain = dataWindNot0.append(dataWind0)
DataTrain.sort_values("datetime", inplace=True)
```

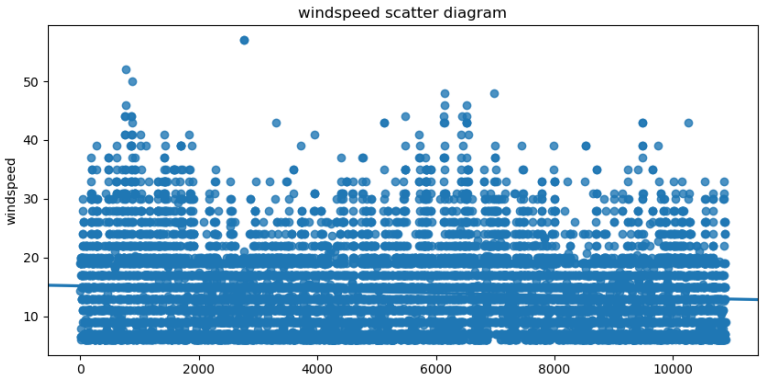


图 9 随机森林处理后的风速分布

2.3 相关分析

经过数据处理后，训练数据集的特征已达到 16 维，但这些特征中并不是所有的特征都有助于建立一个好的预测模型，相反，过多的特征会使得模型发生过拟合，只能在训练集上取得好的结果，而在测试集上表现很差。同时，特征过多也会增加模型训练的时间。为了使得模型有更好的泛化能力，减少模型的训练时间，需要对特征进行合理的特征选择。特征选择前先去除 datetime、casual、registered，因为 datetime 已经进行了数据切割，而测试集中不存在 casual 和 registered。

首先画出相关系数矩阵，得到不同特征之间的关联程度

```
Correlation = DataTrain[:].corr()
mask = np.array(Correlation)
mask[np.tril_indices_from(mask)] = False
fig7 = plt.figure()
fig7.set_size_inches(10, 10)
ax5 = fig7.add_subplot(1, 1, 1)
sns.heatmap(Correlation, mask=mask, square=True, annot=True, cbar=True, ax=ax5)
ax5.set(title="Correlation Analysis")
```

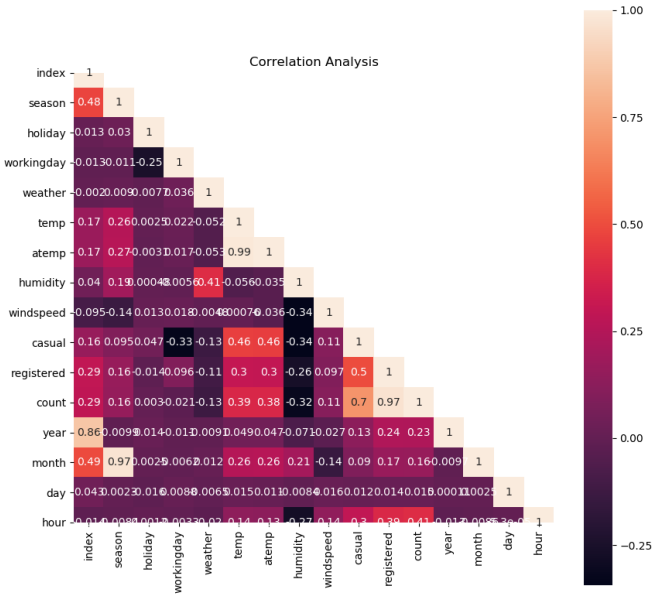


图 10 特征变量相关分析

从图中可以看出颜色越浅，关联程度越高，而 temp 和 atemp 之间的关联程度达到了 0.99，意味着两个特征分布高度重合，进行特征选择时可以相互替代，由于 temp 与预测目标 count 之间的关联程度 0.39>atemp 与 count 之间的关联程度 0.38，最终保留 temp。

最终的预测目标是 count，因此进行特征选择时主要考虑到与 count 之间的关联程度，但关联程度低是否一定意味着彼此影响越小呢？以 hoilday、workingday 与 count 之间的关联度为例，holiday 与 count 之间的关联度为 0.003，workingday 与 count 之间的关联度为-0.021，这两个相关系数非常小，甚至已经可以忽略不记了，但是从日常经验出发这两者对于共享单车租车量应该会有较大影响。当把这两个变量与 hour 组合起来对 count 进行联合分析，发现有所不同：

```
fig8,(ax18,ax19) = plt.subplots(nrows=2)
holiday_hour=
pd.DataFrame(DataTrain.groupby(["hour","holiday"],sort=True)["count"].mean()).reset_index()
sns.pointplot(x=holiday_hour["hour"],     y=holiday_hour["count"],hue=holiday_hour["holiday"],
data=holiday_hour, join=True,ax=ax18)
ax18.set(xlabel='Hour', ylabel='count',title="Count group by Hour Of holiday", label='big')
workingday_hour=
pd.DataFrame(DataTrain.groupby(["hour","workingday"],sort=True)["count"].mean()).reset_inde
x()
sns.pointplot(x=workingday_hour["hour"],
y=workingday_hour["count"],hue=workingday_hour["workingday"],data=workingday_hour,
join=True,ax=ax19)
ax19.set(xlabel='Hour', ylabel='count',title="Count group by Hour Of workingday",label='big')
```

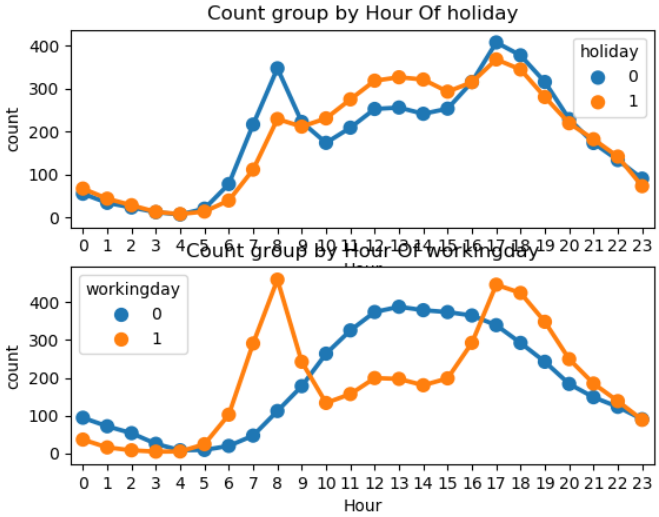


图 11 联合相关度分析

显然 holiday 和 workingday 对于 count 值的分布有较大影响，因此从单变量分析到多变

分析，最终选择的特征 season、holiday、workingday、weather、temp、humidity、windspeed、month、year、hour。

2.4 评价标准

在建立模型前，为了衡量模型的预测效果，我们需要先对模型的性能设定评价标准，kaggle 中 Bike Sharing Demand 比赛提供了评价标准为均方根对数误差 RMSLE：

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

均方根对数误差 RMSLE 是回归问题中经常用到的评价标准，它建立在均方根误差 RMSE 的基础上，但当预测的值的范围很大的时候，RMSE 会容易被一些较大的值主导，也就是说即使有很多较小的值预测正确，只要有少数较大的值预测不准确，RMSE 仍然会很大。于是产生了均方对数误差 RMSLE，它会对数据先取 log 然后再求 RMSE，这样就可以缓解那个问题。下面就是均方根对数误差的函数：

```
def rmsle(y, y_, convertExp=True):
    if convertExp:
        y = np.exp(y),
        y_ = np.exp(y_)
    log1 = np.nan_to_num(np.array([np.log(v + 1) for v in y]))
    log2 = np.nan_to_num(np.array([np.log(v + 1) for v in y_]))
    calc = (log1 - log2) ** 2
    return np.sqrt(np.mean(calc))
```

确定好评价标准后，就可以开始建立回归模型，对数据进行训练和预测。

2.5 模型建立与比较

不同的模型对数据集的预测效果不同，模型之间的相互比较有利于找到更适合的模型。这里采用了三种模型进行对比分析，分别是线性回归模型（Linear Regression）、随机森林回归模型（Random Forest）以及梯度提升模型（Gradient Boosting），分别从 sklearn 库中导入这三个函数，建立对应的模型。

```
# 1.Linear Regression model
L_Model = LinearRegression()
yLabelsLog = np.log1p(yLabels)
L_Model.fit(X=DataTrain, y=yLabelsLog)
preds = L_Model.predict(X=DataTrain)
print("RMSLE Value For Linear Regression: ", rmsle(np.exp(yLabelsLog), np.exp(preds), False))
# 2.random forest model
R_Model = RandomForestRegressor(n_estimators=100)
yLabelsLog = np.log1p(yLabels)
```

```
R_Model.fit(DataTrain,yLabelsLog)
preds = R_Model.predict(X=DataTrain)
print("RMSLE Value For Random Forest: ",rmsle(np.exp(yLabelsLog),np.exp(preds),False))
# 3.Ensemble Model - Gradient Boost
G_Model = GradientBoostingRegressor(n_estimators=4000, alpha=0.01)
yLabelsLog = np.log1p(yLabels)
G_Model.fit(DataTrain, yLabelsLog)
preds = G_Model.predict(X=DataTrain)
print("RMSLE Value For Gradient Boost: ", rmsle(np.exp(yLabelsLog), np.exp(preds), False))
```

运行程序得到三种模型的均方根对数误差 RMSLE 如下图：

```
RMSLE Value For Linear Regression:  0.9784996530291654
RMSLE Value For Random Forest:  0.11387745871357467
RMSLE Value For Gradient Boost:  0.21303416980997197
```

图 12 三种模型的均方根对数误差

由此可见，针对该训练集建立的模型中，随机森林回归模型具有最好的预测效果，RMSLE=0.11,因此最终决定采用随机森林回归模型（Random Forest）。

2.6 模型预测

根据上述建立的随机森林回归模型，对测试集进行测试，并将测试结果输出。

```
predsTest = R_Model.predict(X=DataTest)
submission = pd.DataFrame({
        "datetime": datetimecol,
        "count": [max(0, x) for x in np.exp(predsTest)]
    })
submission.to_csv(r'../MS/bike1_predict.csv', index=False)
Data=pd.read_csv(r'bike_prediction.csv')
Data.head
```

截取某一天的共享单车需求预测数据

2011-01-20 05:00:00	6.525284	2011-01-20 17:00:00	201.044171
2011-01-20 06:00:00	36.961609	2011-01-20 18:00:00	179.480734
2011-01-20 07:00:00	90.431993	2011-01-20 19:00:00	107.505350
2011-01-20 08:00:00	204.861882	2011-01-20 20:00:00	76.547638
2011-01-20 09:00:00	111.866468	2011-01-20 21:00:00	46.767781
2011-01-20 10:00:00	57.015489	2011-01-20 22:00:00	37.661043
2011-01-20 11:00:00	62.037472	2011-01-20 23:00:00	24.448187
2011-01-20 12:00:00	75.422684	2011-01-21 00:00:00	11.047479
2011-01-20 13:00:00	73.444014	2011-01-21 01:00:00	5.631827
2011-01-20 14:00:00	74.618259	2011-01-21 02:00:00	4.173014
2011-01-20 15:00:00	76.941249	2011-01-21 03:00:00	3.126503
2011-01-20 16:00:00	91.727672	2011-01-21 04:00:00	2.918113

图 13 共享单车租车量分布

从上图可以看到 08:00:00-09:00:00,17:00:00-18:00:00 是共享单车租用的两个高峰期，而这两个时间段处于上下班的高峰期，其余时刻的租车量围绕这两个峰值往两端递减，这样的租车分布符合我们日常的认知习惯

三．课题总结

本次作业距离模式识别课程结课有一段时间了，学习模式识别这门课时我就感觉到这门课的难度，虽然赵老师已经尽可能把课程讲的通俗易懂，但初次接触模式识别的我还是感受到了压力，好在时间的投入还是有效果的，对于原理性的知识有了一定的了解。但通过这次作业我发现从理论到实践还有很多路要走。

做这次作业前，我并没有用过 kaggle 平台，虽然有一定的 python 基础，但是也比较薄弱，对于很多跟模式识别相关的库比较陌生。选择共享单车预测这个题目主要出于对这个题目本身的兴趣，做这个题目前我的想法是尽可能把课堂中学到的知识应用到这次作业中来，但很快我就发现有点高估自己，自己现有的 python 编程能力不足以支撑自己的想法，于是我参考了很多在这个比赛中的 Notebook，但给我的感受是大部分并没有做到有理有据，比如为什么选择了这些特征进行预测，理由不够充分，而且最终留下的特征太多。我尝试进行特征融合，尤其是在做数据可视化分析时看到 workingday 本身与 count 的相关性并不高，但是一旦具体与 hour 结合起来就有很明显的差别，说明这个特征不能简单的去除，然而我在特征融合的尝试中发现这会对其他的特征，比如 holiday 有较大影响，只好被迫停止这个尝试。所以虽然本次作业已经算完成，但同样也是一个开始，在后面的学习中，我会不断提高自己的编程能力以及把理论与实践相结合的能力，相信到时候我会比现在做的好。

附：文件说明

上传至 GitHub 的文件包括：

1. 训练集:train.csv
2. 测试集:text.csv
3. 测试结果:bike1_predict.csv
4. 程序:Bike Sharing.py
5. 报告: Bike sharing prediction.pdf