REPORT

BING LIU

Abstract. In this report, I will talk about my work achievement. After systematically learning latex and Git, mastering its basic operations, and at the same time completing python learning, and I test the learning results by completing a predictive kaggle question. The following is a detailed introduction about the kaggle competition of Bike Sharing Demand.

Contents

Date: 2021-04-26.
2020 *Mathematics Subject Classification.* Artificial Intelligence.
Key words and phrases. Machine Learning, Data Mining, python, latex, git.

1. Introduction

1.1. **Background.** The bicycle sharing system is a way of renting bicycles. Through the network of kiosk locations throughout the city, membership is automatically obtained, and the process of renting and returning bicycles. Using these systems, people can rent a bicycle from one place and return it to other places as needed. Currently, there are more than 500 bike sharing programs worldwide.

1.2. **Target.** 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.

2. Data

2.1. **Data Description.** The competition provide hourly rental data spanning two years.the training set is comprised of the first 19 days of each month, while the test set is the 20th to the end of the month. The taskis to predict the total count of bikes rented during each hour covered by the test set, using only information available prior to the rental period.

- **train.csv** – it contains 10886 rows and 12 columns. Each row represents bike rental data for a certain hour. Each column indicates the current conditions
- **test.csv** – it contains 6493 rows and 9 columns. Compared with the train data, there are fewer ”casual”,”registered” and ”count” columns.
- **sampleSubmission.csv** – it clarifies the data submission format. It just contains 2 columns that is ”datetime” and ”count”.

2.2. **Data Fields.** The following is a detailed introduction of the data for each attributes.

column	**description**
datetime	hourly date + timestamp
season	1 = spring, 2 = summer, 3 = fall, 4 = winter
holiday	whether the day is considered a holiday
workingday	whether the day is neither a weekend nor holiday
weather	1=clear, 2=mist + cloudy, 3=light snow, 4=heavy rain
temp	temperature in Celsius
atemp	”feels like” temperature in Celsius
humidity	relative humidity
windspeed	wind speed
casual	number of non-registered user rentals initiated
registered	number of registered user rentals initiated
count	number of total rentals

2.3. **Missing Values Analysis.** I use ”missingno” to visualize missing value in the dataset, Luckily the dataset do not has any missing value.

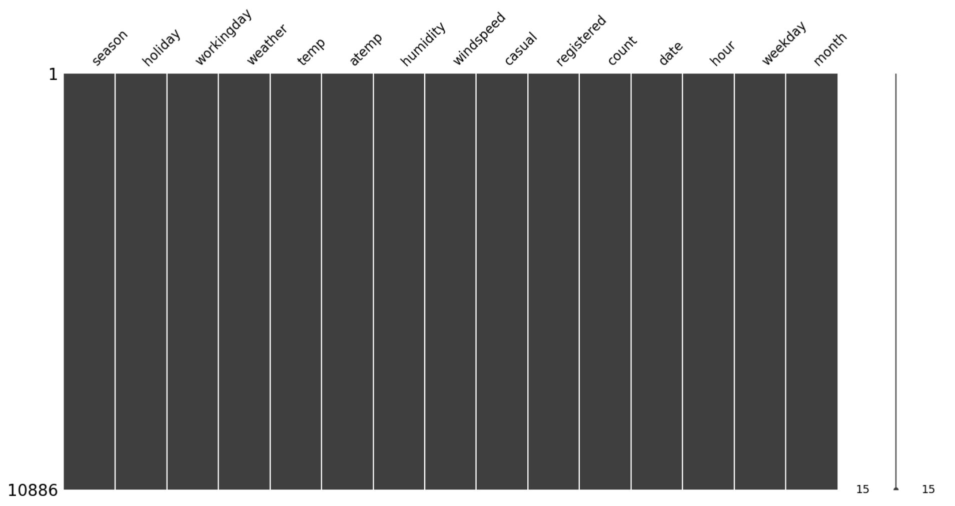


FIGURE 1. Missing values analysis

2.4. **Outliers Analysis.** 1:Spring season has got relatively lower count.
2:The boxplot with "Hour Of The Day" is quiet interesting.The median value are relatively higher at 7AM to 8AM and 5PM to 6PM.
3:Most of the outlier points are mainly contributed from "Working Day" than "Non Working Day".

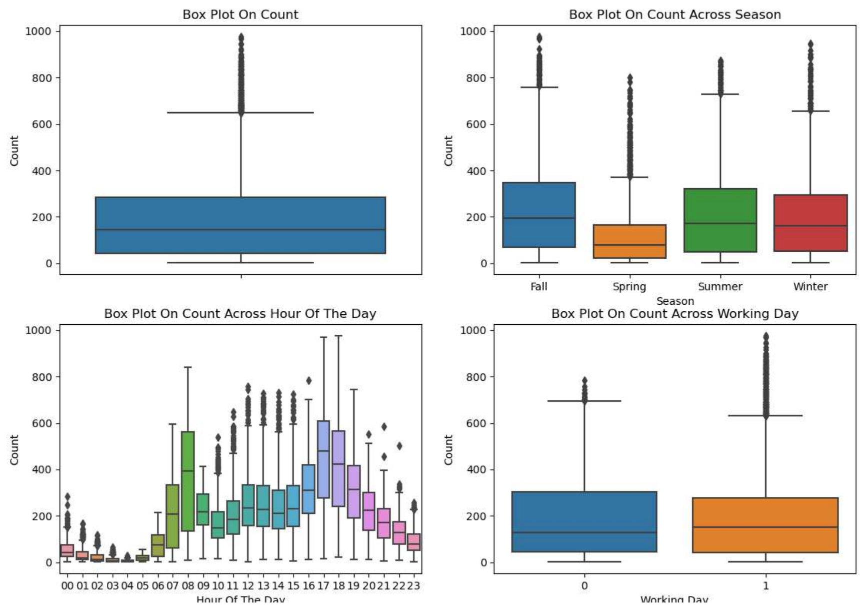


FIGURE 2. Outliers analysis

2.5. **Correlation Analysis.** 1:temp and humidity features has got positive and negative correlation with count respectively. the count variable has got little dependency on ”temp” and ”humidity”.
2:”Casual” and ”Registered” are also not taken into account since they are leakage variables in nature and need to dropped during model building.
3:windspeed is not gonna be really useful numerical feature.

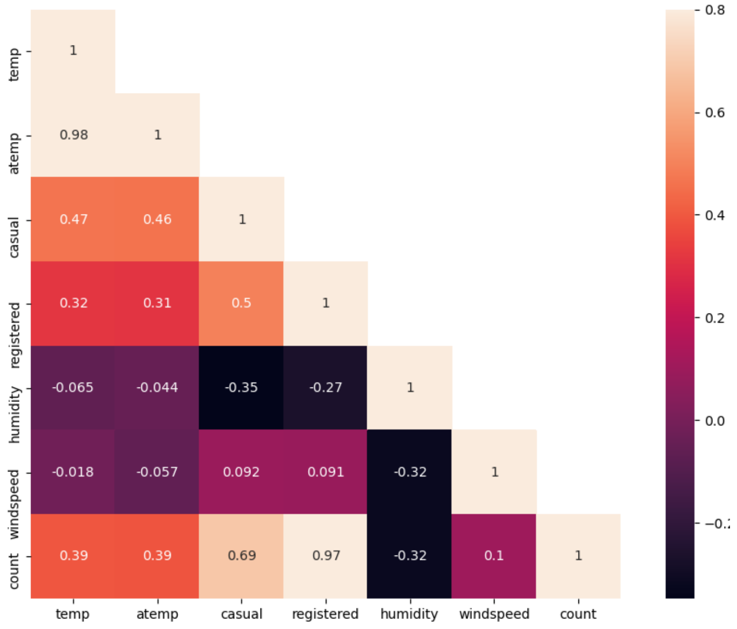


FIGURE 3. Correlation Analysis I

Regression plot in seaborn is one useful way to depict the relationship between two features. Here we consider ”count” vs ”temp”, ”humidity”, ”windspeed”.

2.6. **Visualizing Distribution Of Data.** It is desirable to have Normal distribution as most of the machine learning techniques require dependent variable to be Normal. One possible solution is to take log transformation on ”count” variable after removing outlier data points.

2.7. **Visualizing Count.** 1:It is quiet obvious that people tend to rent bike during summer season. Therefore June, July and August has got relatively higher demand for bicycle.
2:On weekdays more people tend to rent bicycle around 7AM-8AM and 5PM-6PM.
3:On ”Saturday” and ”Sunday”.More people tend to rent bicycle between 10AM and 4PM.
4:Registered user contribute the peak around 7AM-8AM and 5PM-6PM.

3. Feature Engineering and Model

3.1. **Feature Processing.** Split the given date into ”date, hour, year, weekday, month”. According to visual analysis, select features that have strong correlation with count.

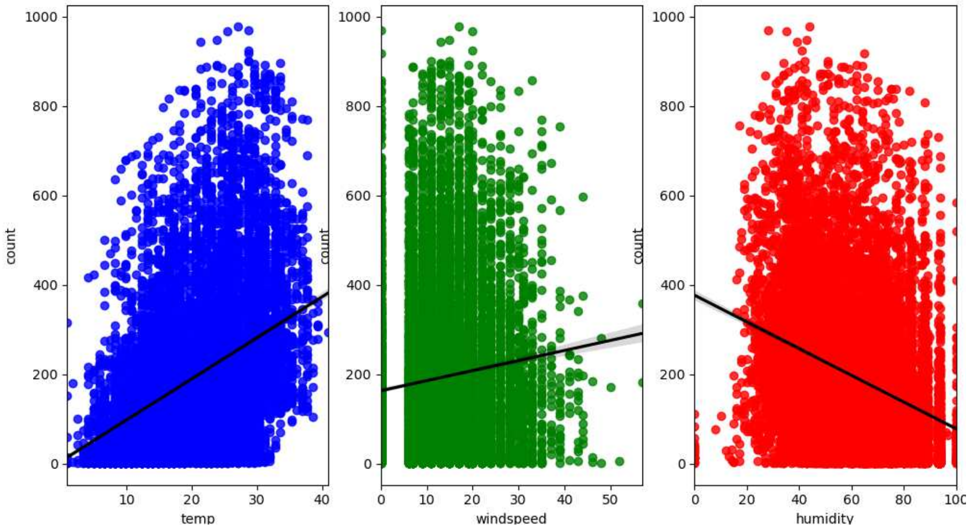


FIGURE 4. Correlation Analysis II

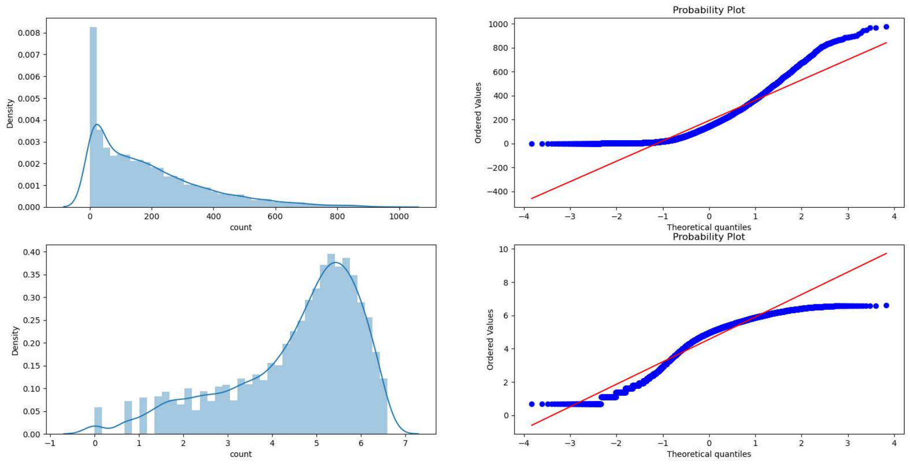


FIGURE 5. Visualizing Distribution Of Data

3.2. **Splitting train and test date.** Divide train set and test set according to whether there is count attribute.

3.3. **Model.** I have choose the Ensemble Model - Gradient Boost. Compare the distribution of train and test results.It confirms visually that the model has not predicted really bad and not suffering from major overfitting problem.

4. CONCLUSIONS

Using RMSLE to calculate the error, it penalizes under-prediction even more.
RMSLE Value For Gradient Boost: 0.189973542608
The score of my submission in kaggle is 0.41867. Ranked 428 among 3242 teams.

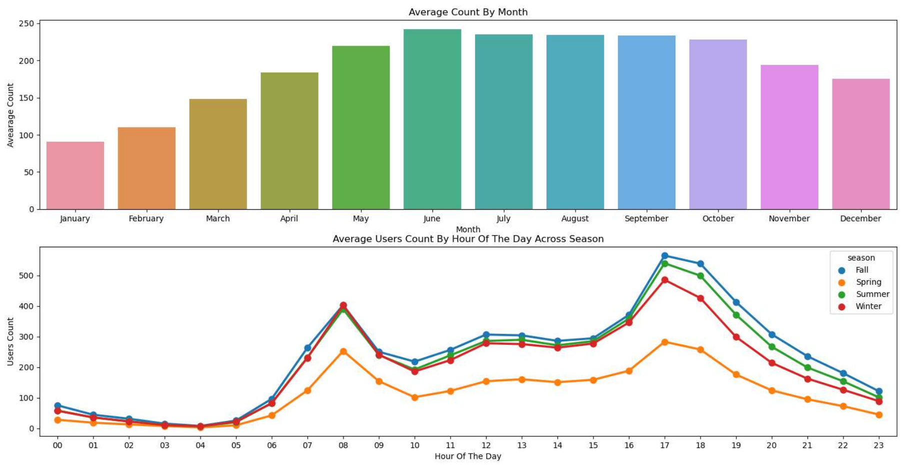


FIGURE 6. Visualizing Count I

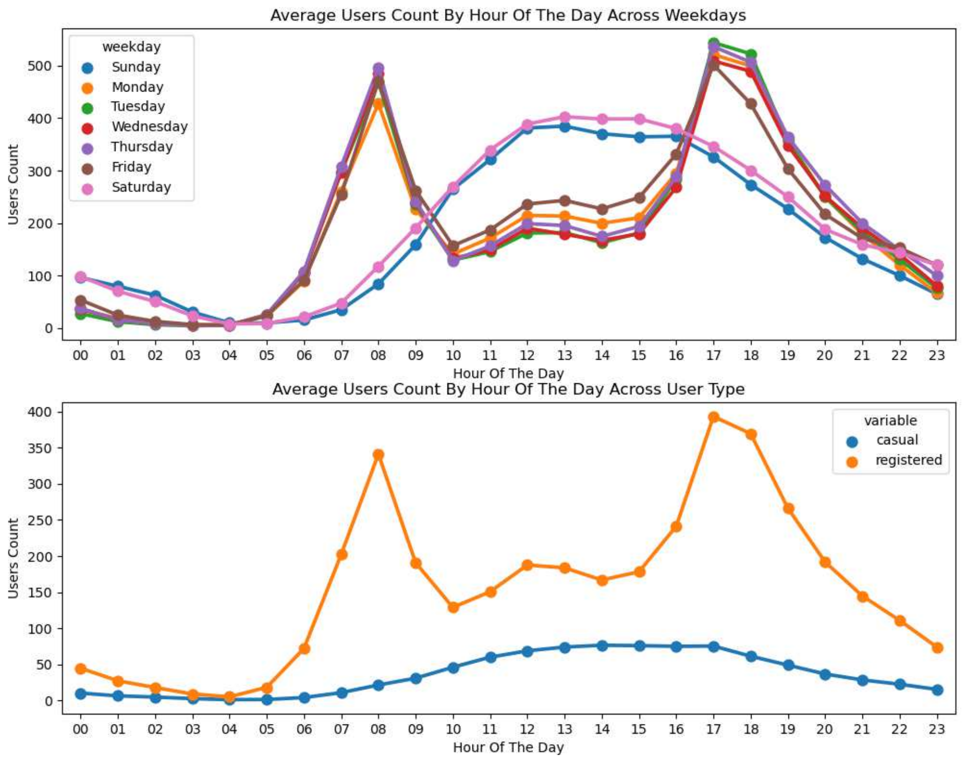
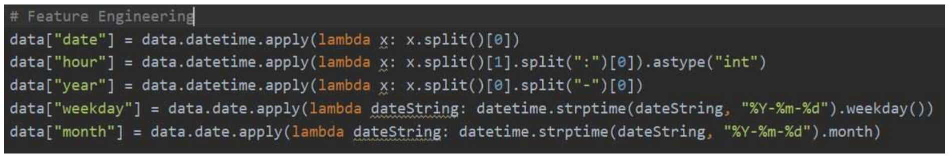


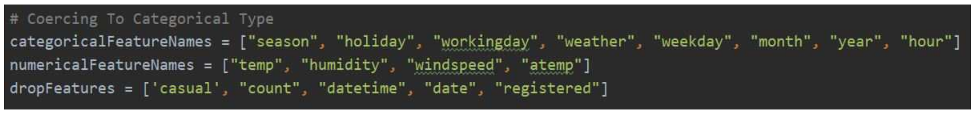
FIGURE 7. Visualizing Count II

As the result shows the model can effectively predict the count demand of bike. By solving this prediction problem, I am more familiar with python visualization and have a guiding idea for the data processing process. The learning task of the first stage was completed well.



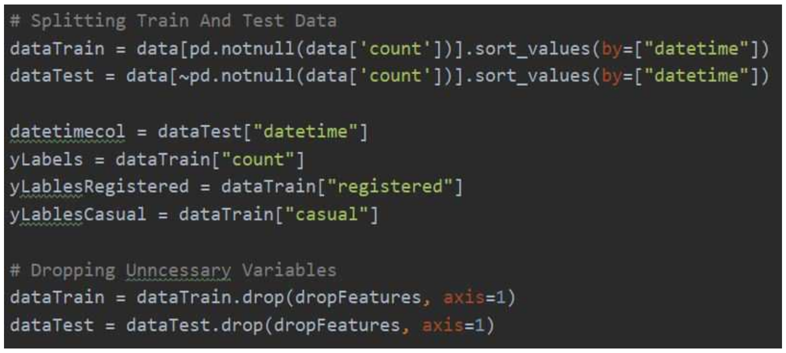
```
# Feature Engineering
data["date"] = data.datetime.apply(lambda x: x.split()[0])
data["hour"] = data.datetime.apply(lambda x: x.split()[1].split(":")[0]).astype("int")
data["year"] = data.datetime.apply(lambda x: x.split()[0].split("-")[0])
data["weekday"] = data.date.apply(lambda dateString: datetime.strptime(dateString, "%Y-%m-%d").weekday())
data["month"] = data.date.apply(lambda dateString: datetime.strptime(dateString, "%Y-%m-%d").month)
```

FIGURE 8. Time feature processing



```
# Coercing To Categorical Type
categoricalFeatureNames = ["season", "holiday", "workingday", "weather", "weekday", "month", "year", "hour"]
numericalFeatureNames = ["temp", "humidity", "windspeed", "atemp"]
dropFeatures = ['casual', "count", "datetime", "date", "registered"]
```

FIGURE 9. Feature selection



```
# Splitting Train And Test Data
dataTrain = data[pd.notnull(data['count'])].sort_values(by=["datetime"])
dataTest = data[~pd.notnull(data['count'])].sort_values(by=["datetime"])

datetimecol = dataTest["datetime"]
yLabels = dataTrain["count"]
yLablesRegistered = dataTrain["registered"]
yLablesCasual = dataTrain["casual"]

# Dropping Unncessary Variables
dataTrain = dataTrain.drop(dropFeatures, axis=1)
dataTest = dataTest.drop(dropFeatures, axis=1)
```

FIGURE 10. Training set and test set division

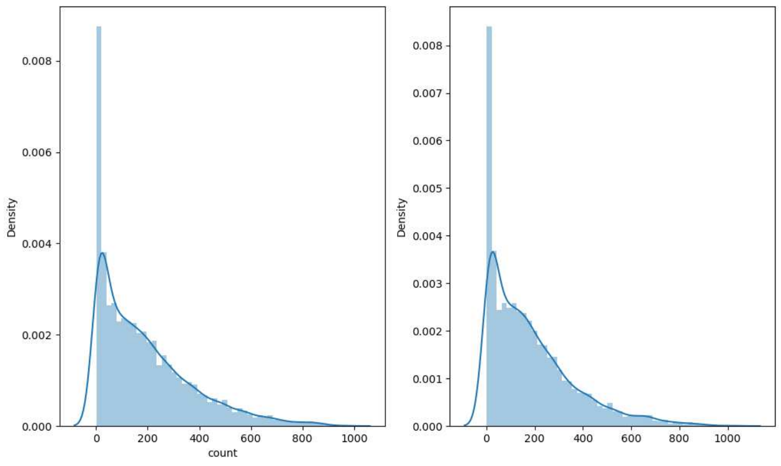


FIGURE 11. Distribution of train and test results

(A. 1) SCHOOL OF COMPUTER SCIENCE,, JILIN UNIVERSITY, CHANGCHUN 130012, CHINA
Email address, A. 1: bliu@tulip.academy