Data Mining & Discovery

Fang Jie

Li Zhiyuan

Group 9
Winter Is Coming

Contents

- Data Analysis
- Preprocessing
- Training
- Improvement

Data analysis

- Data Structure
- Data Visualization

Data Structure

```
training.info()
```

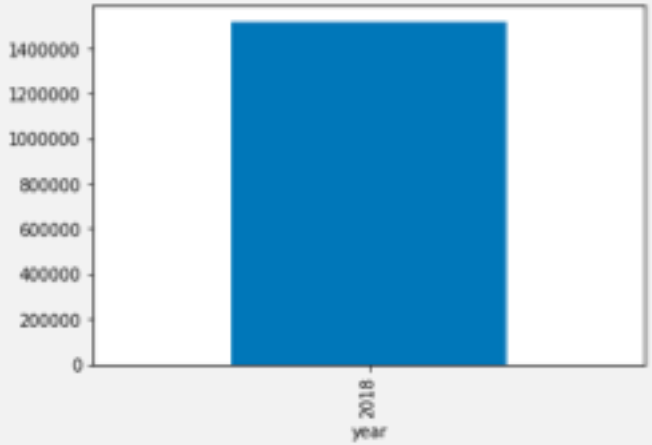
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1513570 entries, 0 to 1513569
Data columns (total 12 columns):
trip_id               1513570 non-null int64
start_time            1513570 non-null datetime64[ns]
end_time              1513570 non-null object
bikeid                1513570 non-null int64
tripduration          1513570 non-null object
from_station_id       1513570 non-null int64
from_station_name     1513570 non-null object
to_station_id         1513570 non-null int64
to_station_name       1513570 non-null object
usertype              1513570 non-null object
gender                1218574 non-null object
birthyear             1221990 non-null float64
dtypes: datetime64[ns](1), float64(1), int64(4), object(6)
memory usage: 138.6+ MB
```

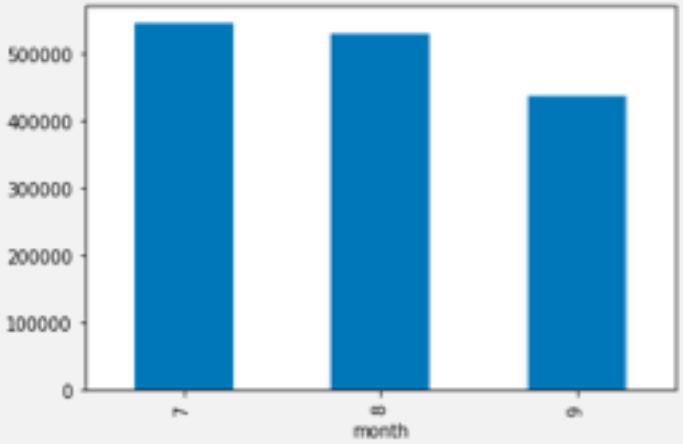
```
training.head(10)
```

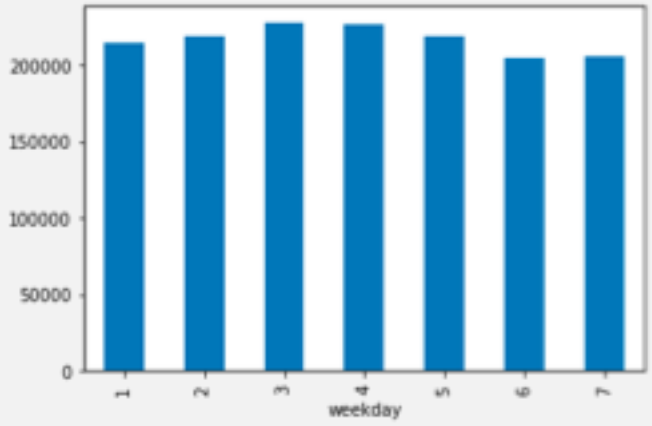
	trip_id	start_time	end_time	bikeid	tripduration	from_station_id	from_station_name	to_station_id	to_station_name	usertype	gender	birthyear
0	19244622	2018-07-01 00:00:03	2018-07-01 23:56:11	5429	86,168.0	140	Dearborn Pkwy & Delaware Pl	106	State St & Pearson St	Customer	NaN	NaN
1	19244623	2018-07-01 00:00:13	2018-07-01 00:06:39	93	386.0	153	Southport Ave & Wellington Ave	250	Ashland Ave & Wellington Ave	Subscriber	Male	1986.0
2	19244624	2018-07-01 00:00:15	2018-07-01 00:23:26	2461	1,391.0	76	Lake Shore Dr & Monroe St	301	Clark St & Schiller St	Subscriber	Female	1987.0
3	19244625	2018-07-01 00:00:25	2018-07-01 00:23:31	2991	1,386.0	76	Lake Shore Dr & Monroe St	301	Clark St & Schiller St	Subscriber	Male	1986.0
4	19244626	2018-07-01 00:00:27	2018-07-01 00:11:23	2851	656.0	60	Dayton St & North Ave	166	Ashland Ave & Wrightwood Ave	Subscriber	Male	1961.0
5	19244627	2018-07-01 00:00:35	2018-07-01 00:16:09	5980	934.0	128	Damen Ave & Chicago Ave	71	Morgan St & Lake St	Subscriber	Male	1995.0
6	19244628	2018-07-01 00:00:37	2018-07-01 00:10:14	3132	577.0	168	Michigan Ave & 14th St	321	Wabash Ave & 9th St	Customer	NaN	NaN
7	19244629	2018-07-01 00:00:55	2018-07-01 00:09:20	2281	505.0	168	Michigan Ave & 14th St	321	Wabash Ave & 9th St	Customer	NaN	NaN
8	19244630	2018-07-01 00:01:38	2018-07-01 00:25:25	3465	1,427.0	229	Southport Ave & Roscoe St	324	Stockton Dr & Wrightwood Ave	Customer	NaN	NaN
9	19244631	2018-07-01 00:01:44	2018-07-01 00:25:25	3873	1,421.0	229	Southport Ave & Roscoe St	324	Stockton Dr & Wrightwood Ave	Customer	NaN	NaN

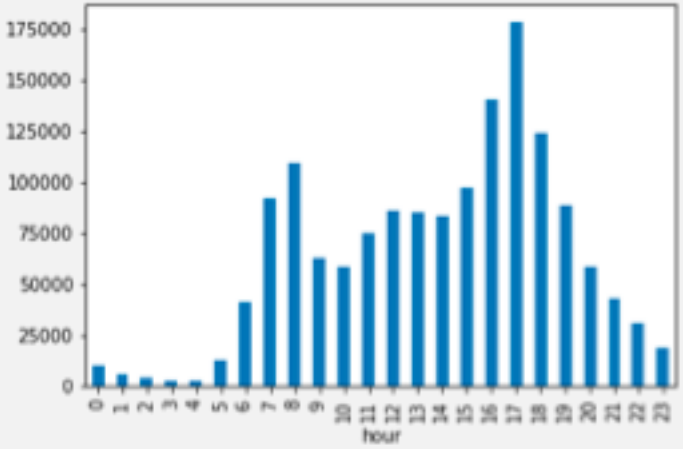
Data Visualization

Time Visualization



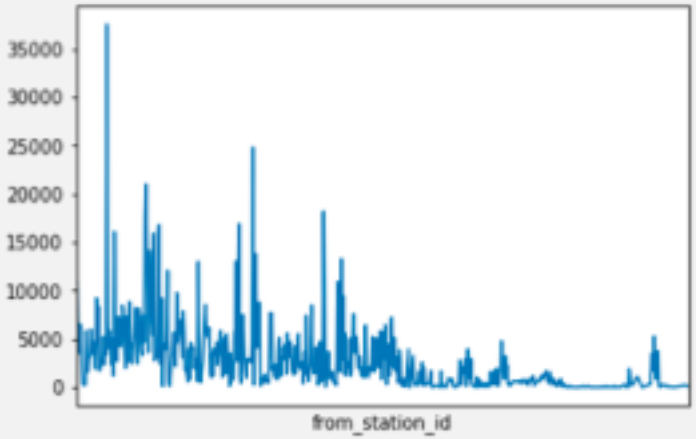


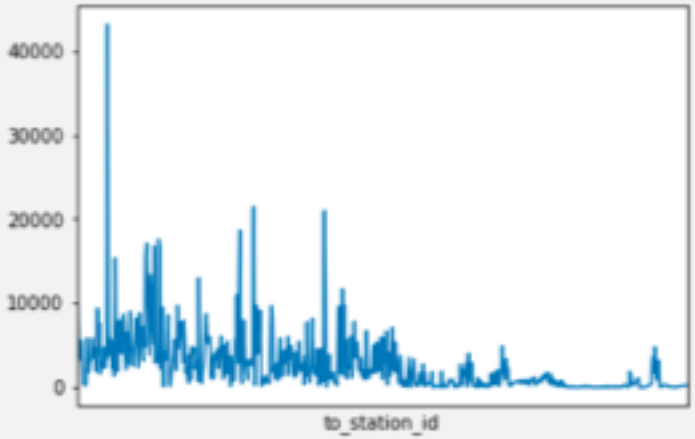




Data Visualization

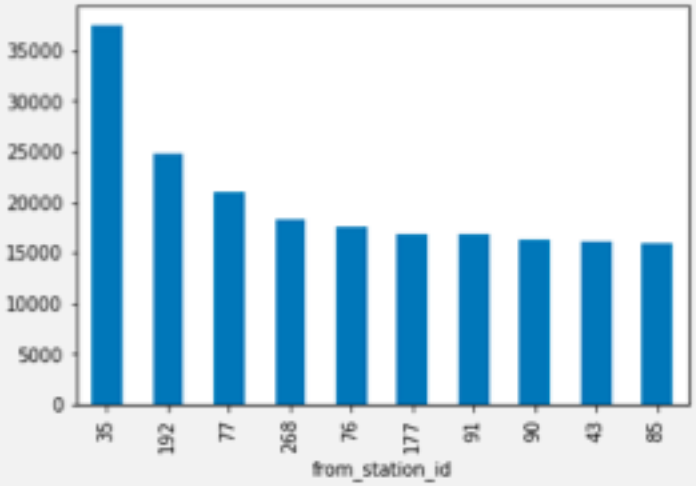
Station Visualization

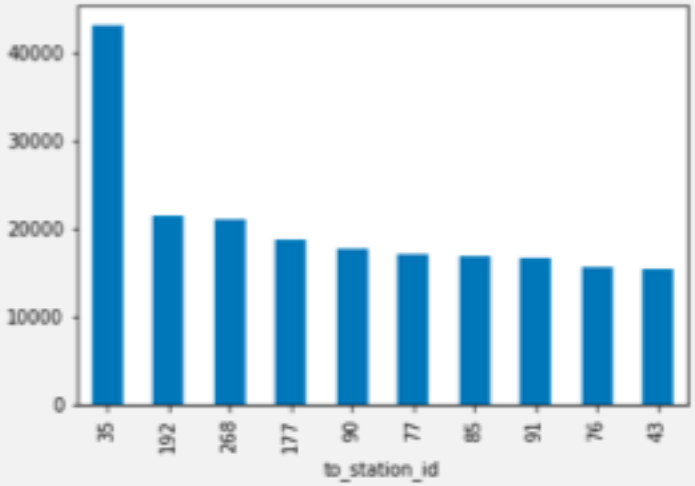




Top 10 popular start station

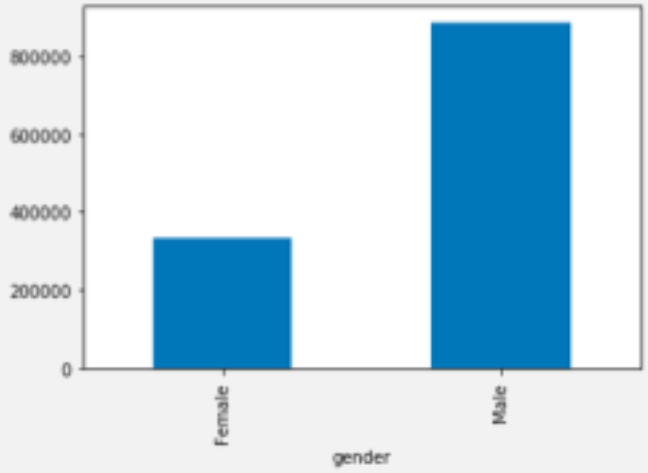
Top 10 popular end station

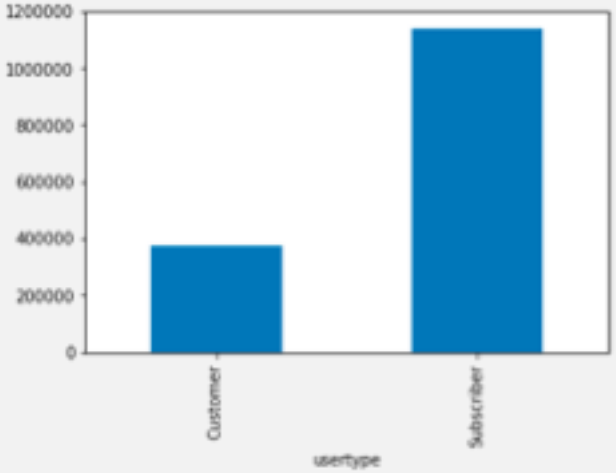




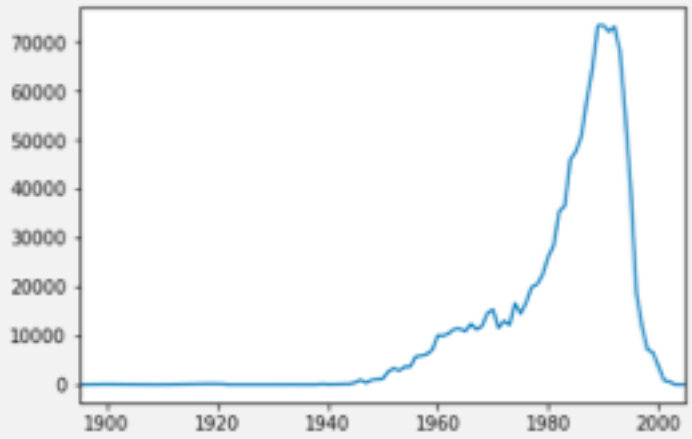
Data Visualization

Other Visualization





birthyear



Tripduration statistic

count	1.513570e+06
mean	2.671152e+01
std	6.310889e+02
min	1.000000e+00
25%	7.000000e+00
50%	1.200000e+01
75%	2.300000e+01
max	1.939160e+05

Preprocessing

- Data Reduction
- Data Transformation
- Data Cleaning
- Data Integration

Data Reduction

Delete Attributes

We delete 'end_time', 'bikeid', 'from_station_name', 'to_station_name'

```
# delete the useless data
training_data.drop(['trip_id', 'end_time', 'bikeid', 'from_station_name', 'to_station_name']\
                   , axis = 1, inplace = True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1513570 entries, 0 to 1513569
Data columns (total 7 columns):
start_time          1513570 non-null datetime64[ns]
tripduration        1513570 non-null object
from_station_id     1513570 non-null int64
to_station_id       1513570 non-null int64
usertype            1513570 non-null object
gender              1218574 non-null object
birthyear           1221990 non-null float64
dtypes: datetime64[ns](1), float64(1), int64(2), object(3)
memory usage: 80.8+ MB
```

Data Transformation

- Convert non-numeric data to be numeric
- Convert birthyear to age
- Age Discretization

```
training_data['usertype'] = training_data['usertype'].map(lambda x: -1 if x == 'Customer' else 1)
training_data['gender'] = training_data['gender'].map(lambda x: 1 if x == 'Male' else (-1 if x == 'Female' else 0))
training_data['birthyear'] = training_data['birthyear'].map(lambda x: int((2018 - x - 1) / 5) if isnan(x)is not True else 0)
training_data['age'] = training_data['birthyear'].map(lambda x: x if x >= 0 and x <= 15 else 16)
training_data['month'] = training_data['start_time'].dt.month
training_data['weekday'] = training_data['start_time'].dt.dayofweek+1
training_data['start_time'] = training_data['start_time'].dt.hour
training_data['tripduration'] = training_data['tripduration'].map(lambda x: int(float(x.strip().replace(',', '')) / 60))
```

```
training_data.drop(['birthyear'],axis = 1,inplace = True)
```

```
training_data.head(10)
```

	start_time	tripduration	from_station_id	to_station_id	usertype	gender	age	month	weekday
0	0	1436	140	106	-1	0	0	7	7
1	0	6	153	250	1	1	6	7	7
2	0	23	76	301	1	-1	6	7	7
3	0	23	76	301	1	1	6	7	7
4	0	10	60	166	1	1	11	7	7
5	0	15	128	71	1	1	4	7	7
6	0	9	168	321	-1	0	0	7	7
7	0	8	168	321	-1	0	0	7	7
8	0	23	229	324	-1	0	0	7	7
9	0	23	229	324	-1	0	0	7	7

Data Cleaning

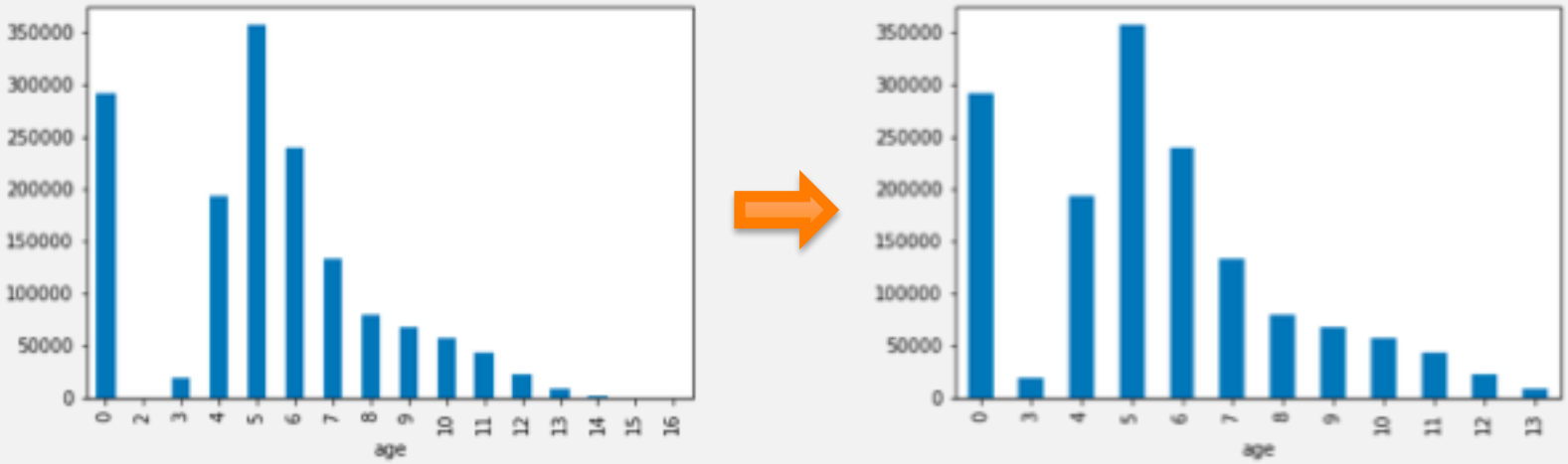
Delete noise that duration>600

```
training_data['tripduration'].describe(percentiles=[.5, .9, .95, .99, .999])
```

```
count    1.513570e+06
mean     2.671152e+01
std      6.310889e+02
min      1.000000e+00
50%      1.200000e+01
90%      3.900000e+01
95%      6.300000e+01
99%      1.360000e+02
99.9%    6.488620e+02
max      1.939160e+05
Name: tripduration, dtype: float64
```

```
training_data = training_data[training_data['tripduration'] < 600.0]
```

Delete noise that age is too young or too old



Data Integration

Month Integration

As we have finished data analysis, we know that the number of month in training data only has 7, 8, 9.

So we decide to change the month in test data to 7~9.

For Example, we change 6 to 7, 10 to 9.

```
test_data['month'] = test_data['start_time'].dt.month
test_data['month'] = test_data['month'].map(lambda x: 7 if x<7 else(9 if x>9 else x))
```

```
print(test_data1['month'].head(5))
print(test_data2['month'].head(5))
```

```
0    9
1    9
2    9
3    9
4    9
Name: month, dtype: int64
0    7
1    7
2    7
3    7
4    7
Name: month, dtype: int64
```

Training

GBDT

- Gradient Boosting Decision Tree
- The main idea is it builds decision trees and it generalizes them by allowing optimization of an arbitrary differentiable loss function.
- But it is too slow to run in our weak laptops.

KNN

- The k-nearest neighbors algorithm is among the simplest of all machine learning algorithms.
- An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors
- Time complexity of KNN is O(n)
- Insensitive to abnormal points

Training-KNN

The default parameters for API

class sklearn.neighbors.KNeighborsClassifier(

n_neighbors = 5,
weights = ‘uniform’,
algorithm = ‘auto’,
leaf_size=30,
p = 2,
metric = ‘minkowski’,
metric_params = None,
n_jobs = None,
**kwargs

)

Training-KNN

Results

```
this is the 1 th round
 training accuracy is:  26.580602449520025
 test1 accuracy is:  24.6
 test2 accuracy is:  15.7
this is the 2 th round
 training accuracy is:  27.302217808672623
 test1 accuracy is:  24.5
 test2 accuracy is:  16.400000000000002
this is the 3 th round
 training accuracy is:  27.242634889109567
 test1 accuracy is:  24.099999999999998
 test2 accuracy is:  16.3
this is the 4 th round
 training accuracy is:  26.951340615690167
 test1 accuracy is:  24.7
 test2 accuracy is:  16.400000000000002
```

Training-KNN

Problem

The accuracy is very low, so we want to improve it.

How

- Process the data again.
- Change the training data structure.
- Change the parameters in the training model.

Improvement

- Data improvement
- Model parameters optimization

Data improvement

Original Data

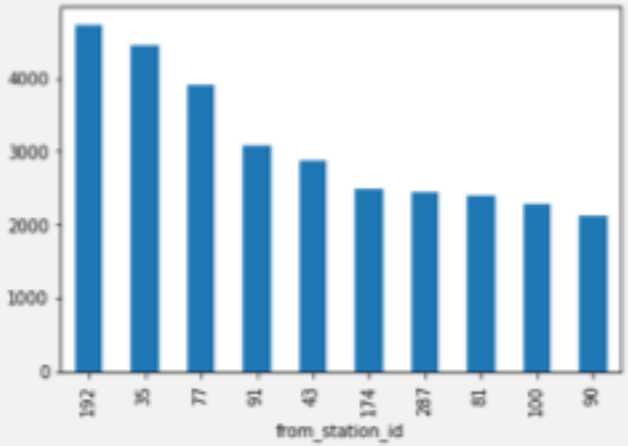
id	start_time	tripduration	from_station_id	to_station_id	usertype	gender	age	month	weekday
0	0	1436	140	106	-1	0	0	7	7
1	0	6	153	250	1	1	6	7	7
2	0	23	76	301	1	-1	6	7	7

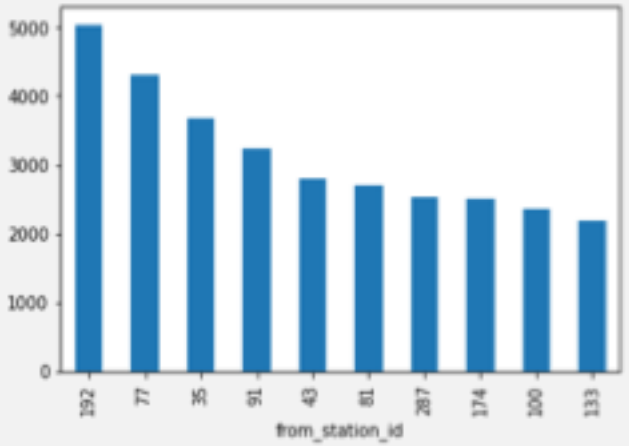
We try to make changes to 4 of these attributes.

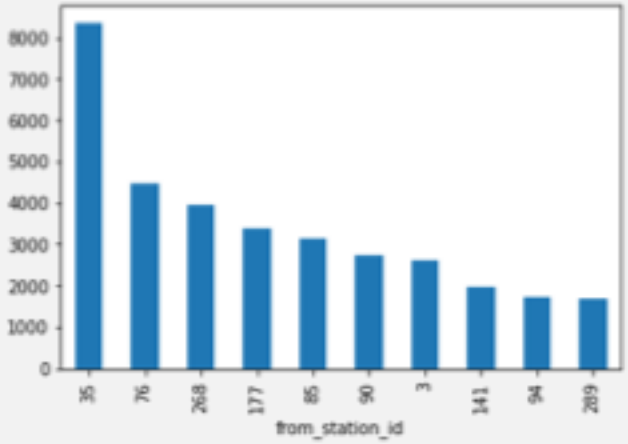
- weekday
- start_time
- tripduration
- age

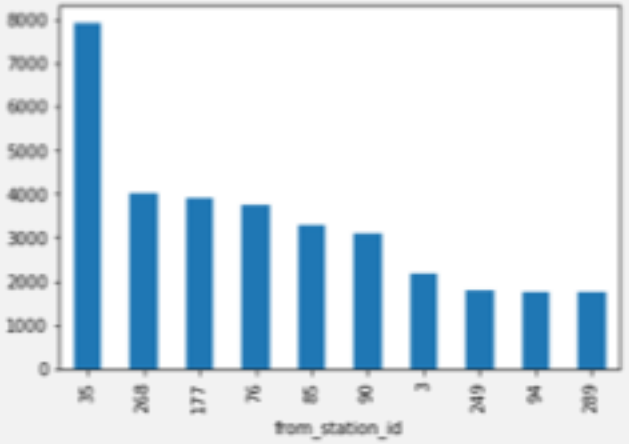
Data improvement

weekday Improvement



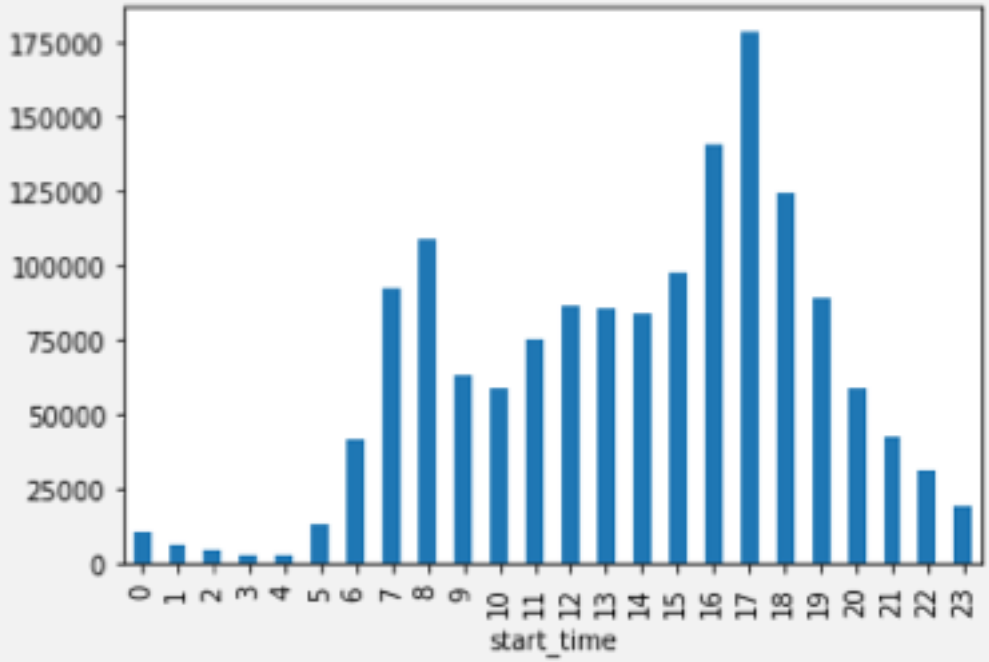






Data improvement

start_time Improvement

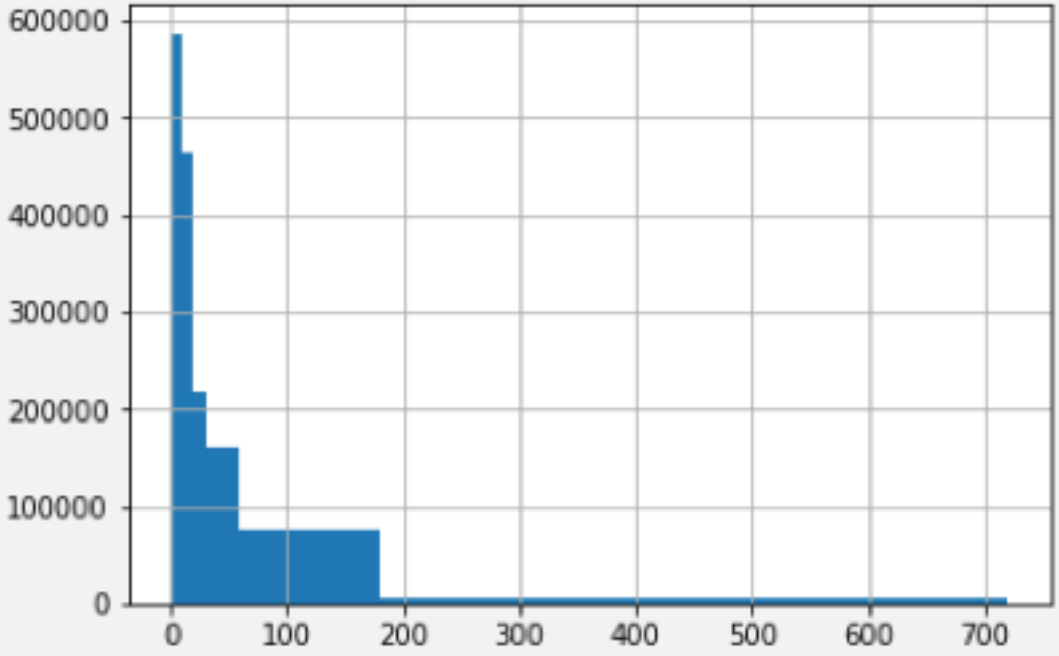



```
training_data['time_range_morning'] = training_data['start_time']\
    .map(lambda x: 1 if x>6 and x<=9 else 0)

training_data['time_range_night'] = training_data['start_time']\
    .map(lambda x: 1 if x>14 and x<=19 else 0)
```

Data improvement

tripduration Improvement

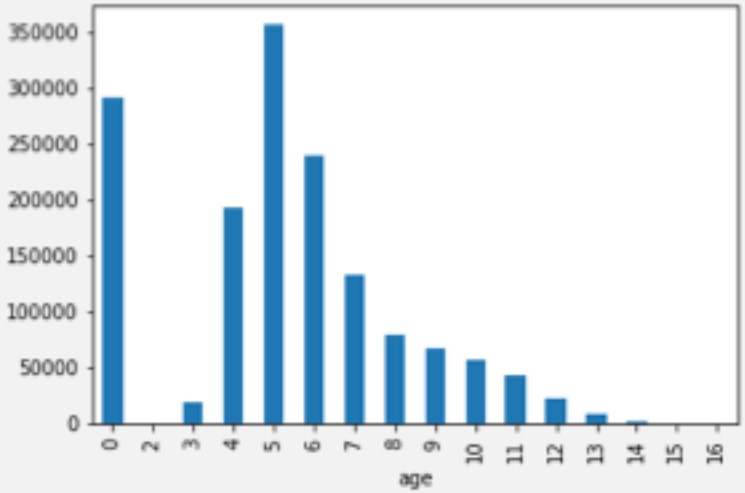


Delete samples with tripduration > 600 mins

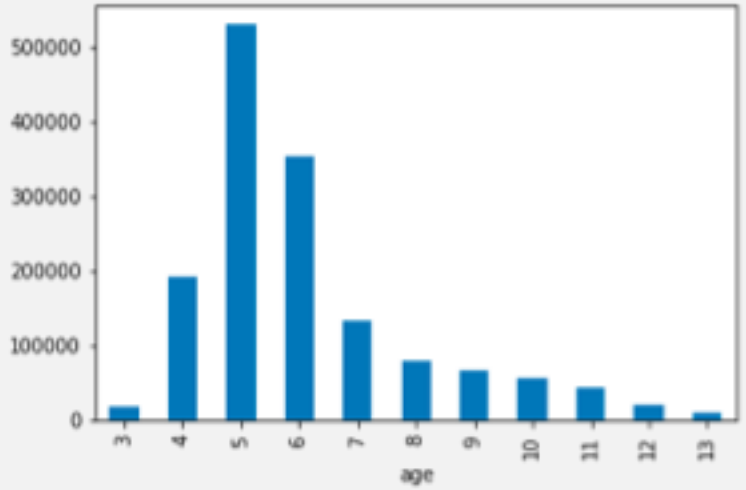
Delete samples with tripduration > 10000 mins

Data improvement

age Improvement



Set age 0 to 5 or 6 by portionally
Probability: 5(60%), 6(40%)



Data improvement

Copy Attribute

And we think that some attributes may have more weight than others.

Through the experiment, we finally decide to copy some of the attributes again.

Final Data

	start_time	tripduration	from_station_id	usertype	gender	age	month	weekend	morning_rush_hour	night_rush_hour	from_station_id	from_station_id	start_time	usertype
0	0	1436	140	-1	0	0	7	1	0	0	140	140	0	-1
1	0	6	153	1	1	6	7	1	0	0	153	153	0	1
2	0	23	76	1	-1	6	7	1	0	0	76	76	0	1
3	0	23	76	1	1	6	7	1	0	0	76	76	0	1
4	0	10	60	1	1	11	7	1	0	0	60	60	0	1

Data improvement

Before improvement

```
this is the 1 th round
 training accuracy is:  26.580602449520025
 test1 accuracy is:  24.6
 test2 accuracy is:  15.7
this is the 2 th round
 training accuracy is:  27.302217808672623
 test1 accuracy is:  24.5
 test2 accuracy is:  16.400000000000002
this is the 3 th round
 training accuracy is:  27.242634889109567
 test1 accuracy is:  24.099999999999998
 test2 accuracy is:  16.3
this is the 4 th round
 training accuracy is:  26.951340615690167
 test1 accuracy is:  24.7
 test2 accuracy is:  16.400000000000002
```

After improvement

```
this is the 1 th round
 training accuracy is:  28.162859980139025
 test1 accuracy is:  25.900000000000002
 test2 accuracy is:  16.900000000000002
this is the 2 th round
 training accuracy is:  27.46110559417411
 test1 accuracy is:  26.1
 test2 accuracy is:  17.7
this is the 3 th round
 training accuracy is:  27.851704733531946
 test1 accuracy is:  26.3
 test2 accuracy is:  17.4
this is the 4 th round
 training accuracy is:  27.851704733531946
 test1 accuracy is:  25.8
 test2 accuracy is:  16.900000000000002
```

Model parameters optimization

sklearn.model_selection.GridSearchCV

Exhaustive search over specified parameter values for an estimator.
There are total 8 parameters in knn, and we try to make change to 5 of them.

- n_neighbors: Number of neighbors to use by default for kneighbors queries.
- weights: weight function used in prediction.
- algorithm: Algorithm used to compute the nearest neighbors:
- leaf_size: Leaf size passed to BallTree or KDTree.
- p : Power parameter for the Minkowski metric.

And the other 3 parameters we decide to use the default value.

Model parameters optimization

Final result

```
##knn model
estimator=KNeighborsClassifier(n_neighbors=9, weights='distance', algorithm='kd_tree', leaf_size=80, p=1)
estimator.fit(X_train,Y_train)
```

```
this is the 1 th round
 training accuracy is:  34.2469380999669
 test1 accuracy is:  31.2
 test2 accuracy is:  21.4
this is the 2 th round
 training accuracy is:  34.207216153591524
 test1 accuracy is:  31.3
 test2 accuracy is:  21.5
this is the 3 th round
 training accuracy is:  33.995365772922874
 test1 accuracy is:  30.599999999999998
 test2 accuracy is:  20.8
this is the 4 th round
 training accuracy is:  34.47864945382324
 test1 accuracy is:  30.7
 test2 accuracy is:  20.7
```

Comments

There are 2 factors leads to such bad performance

- Training set and test set belong to different months
- Number of target stations is too large

THANK YOU FOR WATCHING

Q&A

Group 9
Winter Is Coming