*

Pittsburgh Bike Share Data: Progress Report

AYUSH MISHRA (amm428@pitt.edu) JUNWEI ZHUGE (juz25@pit.edu)

ABSTRACT

The aim of this project is to predict the bike share counts for different stations across the city of Pittsburgh. The raw data is transformed to obtain actual results. In this Progress report, we made use of K-NN, LOGISTIC REGRESSION(LR) and NAÏVE BAYESIAN(NB) methods to train our model and predict the counts. A further aim is also to compare these counts against the rack quantities at each of the bike stations. The predictions based on our models can help determine the optimum time matrix to "re-balance" the bike accumulation and restore them to their optimum levels across the city.

1. INTRODUCTION

In this report, we first combine the Q3 and Q4 bike share data into a single entity to ease the processing. This gives us a consolidated view of the data and we can work on this to give us a better view of the response variable that we need to predict. At this point however, we do not have the actual response variable. We further process the dataset to get rid of "missing values". After this, we split some consolidated columns such as "StartTime" and "StopTime into more edifying and workable columns.

Our goal is to predict the counts that are *rented out* and are *deposited in* to each of the stations each hour. Thus, we move one step forward by splitting the above mentioned consolidated columns into more processible data such as "StartMonth", "StartDay", "StartHour", "StopMonth"…etc. This brings us closer to determining the response variable.

We aggregate the dataset by grouping together predictor variables and determining the counts. This count data is our response variable. The detailed methods of our predictions are shown in the following sections.

2. METHODS

For the initial processing we choose to apply K-NN(1,5,10), LR and NB methods to show the contrasts in the predicted data. We separate the counts to the ones that "Start" at a station and the ones that "Stop" at the stations. We merge these datasets to create a total counts dataset. We then create the training and test sets for predictors. These methods give us a peek at the prediction of the response counts as described in the following:

2.1 LOGISTIC REGRESSION

We extract the count data for the number of bikes *rented out* of a station named as the start station count ("StartCount") and the number of bikes *deposited in* to a station named as station stop count ("StopCount"). The actual response variable for the model predictors are determined by the difference in start counts and stop counts each hour. This gives us a clear picture of the residual number of bikes at a given station ("Count"). This *Count* variable can have a positive or a negative value (*StopCount – StartCount*). A positive value means more number of bikes deposited in a particular hour than rented out. Negative value means more number of rentals than deposits. A zero value means *rentals equal deposits* for the particular hour.

The Logistic regression Model analyzes the *Count* response variable against all other dependent factors. Here, the predictor variables are "FromStationId", "StartUserType", "StartMonth", "StartDay", "StartHour", "StartRackQnty", "StopUserType", "StopHour". The data is normalized before putting through the process of classifier building. The model uses a probabilistic approach to predict the response variable with a *cutoff probability = 0.5*. The average error in probability prediction is also low, *error = 0.0001620045*. The model displays high Accuracy of results, *Accuracy = 0.999838*. Refer to "Table 1: LR Prediction" for the results. All results are via "k-fold" cross validation scheme with *k = 10*.

2.2 K-NN (K- NEAREST NEIGHBORS)

We analyze the data further with the K-NN analysis model. The results of the analysis with $k = 1$, are almost similar to the logistic regression model with the *error = 0.0002400731*. This is slightly higher than the LR model. Accuracy is also considerably lower, *Accuracy = 0.834844*. This means that the model performs moderately well with the data.

We further analyze the data with *k = 5* and *k = 10*. The results seem to deteriorate with increasing k values with *k = 1* being the best performing model. Refer "Tables 2,3 and 4".

2.3 NAÏVE BAYESIAN

This method does not provide any good results and can be considered as a lower baseline of performances for models. See "Table 5" for the results of the evaluation.

2.4 Tables

We show here the model performances of each of the prediction methods applied.

Table 1: LR Model predicted data

10 -fold CV run lr: #training: 99973 #testing 55554

actual	0	1
0	0	9
0.0227272727272727	0	2
0.0681818181818182	0	1
0.181818181818182	0	9
0.204545454545455	0	2
0.25	0	1
0.318181818181818	5	4
0.340909090909091	2	0
0.386363636363636	1	0
0.409090909090909	8	7
0.431818181818182	12	8
0.454545454545455	9	1
0.477272727272727	9	5
0.5	26	24
0.522727272727273	24	22
0.545454545454545	41	41
0.568181818181818	79	49
0.590909090909091	115	91
0.613636363636364	186	221
0.636363636363636	347	439
0.659090909090909	664	764
0.681818181818182	1286	1926
0.704545454545455	3213	5899
0.727272727272727	6581	21084
0.75	1174	6525
0.772727272727273	256	2145
0.795454545454545	77	1059
0.818181818181818	24	457
0.840909090909091	3	260
0.863636363636364	0	129
0.886363636363636	0	101
0.909090909090909	2	41
0.931818181818182	0	27
0.954545454545455	0	40
0.977272727272727	0	11
1	0	6

Table 2: K-NN Model (k = 1)

10 -fold CV run knn1: #training: 99973 #testing 55554

actual	0	1
0	0	9
0.0227272727272727	0	2
0.0681818181818182	0	1
0.181818181818182	2	7
0.204545454545455	0	2
0.25	1	0
0.318181818181818	6	3
0.340909090909091	1	1
0.386363636363636	1	0
0.409090909090909	5	10
0.431818181818182	4	16
0.454545454545455	1	9
0.477272727272727	5	9
0.5	12	38
0.522727272727273	22	24
0.545454545454545	19	63
0.568181818181818	61	67
0.590909090909091	93	113
0.613636363636364	187	220
0.636363636363636	320	466
0.659090909090909	546	882
0.681818181818182	1181	2031
0.704545454545455	2846	6266
0.727272727272727	6823	20842
0.75	2243	5456
0.772727272727273	745	1656
0.795454545454545	402	734
0.818181818181818	168	313
0.840909090909091	114	149
0.863636363636364	54	75
0.886363636363636	41	60
0.909090909090909	16	27
0.931818181818182	10	17
0.954545454545455	22	18
0.977272727272727	5	6
1	1	5

Table 3: K-NN Model (k = 5)
10 -fold CV run knn5 : #training: 99973 #testing 55554

actual	0	1
0	0	9
0.0227272727272727	0	2
0.0681818181818182	1	0
0.181818181818182	1	8
0.204545454545455	0	2
0.25	0	1
0.318181818181818	6	3
0.340909090909091	1	1
0.386363636363636	1	0
0.409090909090909	9	6
0.431818181818182	5	15
0.454545454545455	3	7
0.477272727272727	9	5
0.5	29	21
0.522727272727273	30	16
0.545454545454545	45	37
0.568181818181818	77	51
0.590909090909091	114	92
0.613636363636364	244	163
0.636363636363636	412	374
0.659090909090909	750	678
0.681818181818182	1557	1655
0.704545454545455	3797	5315
0.727272727272727	9247	18418
0.75	3074	4625
0.772727272727273	1123	1278
0.795454545454545	592	544
0.818181818181818	248	233
0.840909090909091	156	107
0.863636363636364	69	60
0.886363636363636	63	38
0.909090909090909	19	24
0.931818181818182	12	15
0.954545454545455	26	14
0.977272727272727	7	4
1	6	0

Table 4: K-NN Model (k = 10)
10 -fold CV run knn10 : #training: 99973 #testing 55554

actual	0	1
0	2	7
0.0227272727272727	2	0
0.0681818181818182	0	1
0.181818181818182	3	6
0.204545454545455	1	1
0.25	0	1
0.318181818181818	9	0
0.340909090909091	2	0
0.386363636363636	1	0
0.409090909090909	12	3
0.431818181818182	19	1
0.454545454545455	8	2
0.477272727272727	10	4
0.5	40	10
0.522727272727273	40	6
0.545454545454545	64	18
0.568181818181818	100	28
0.590909090909091	162	44
0.613636363636364	306	101
0.636363636363636	580	206
0.659090909090909	1036	392
0.681818181818182	2203	1009
0.704545454545455	5497	3615
0.727272727272727	13322	14343
0.75	4507	3192
0.772727272727273	1568	833
0.795454545454545	808	328
0.818181818181818	350	131
0.840909090909091	215	48
0.863636363636364	102	27
0.886363636363636	84	17
0.909090909090909	28	15
0.931818181818182	24	3
0.954545454545455	35	5
0.977272727272727	9	2
1	5	1

Table 6: Naïve Bayesian Model

10 -fold CV run nb : #training: 99973 #testing 55554

actual	0	1
0	0	0
0.0227272727272727	0	0
0.0681818181818182	0	0
0.181818181818182	0	0
0.204545454545455	0	0
0.25	0	0
0.318181818181818	0	0
0.340909090909091	0	0
0.386363636363636	0	0
0.409090909090909	0	0
0.431818181818182	0	0
0.454545454545455	0	0
0.477272727272727	0	0
0.5	0	0
0.522727272727273	0	0
0.545454545454545	0	0
0.568181818181818	0	0
0.590909090909091	0	0
0.613636363636364	0	0
0.636363636363636	0	0
0.659090909090909	0	0
0.681818181818182	0	0
0.704545454545455	0	0
0.727272727272727	0	0
0.75	0	0
0.772727272727273	0	0
0.795454545454545	0	0
0.818181818181818	0	0
0.840909090909091	0	0
0.863636363636364	0	0
0.886363636363636	0	0
0.909090909090909	0	0
0.931818181818182	0	0
0.954545454545455	0	0
0.977272727272727	0	0
1	0	0

3. FURTHER SCOPE

The analysis shown above is just a preliminary analysis of the data. We can use these models to predict actual counts on an hourly basis and can compare these data to the individual rack quantities at each of the stations and thus can determine the most appropriate times to "re-balance" the bikes for efficient bike sharing across the city.

In order to make this efficient, we shall explore further number of methods such as SVM, DECISION TREES and ADA.

For example, the K-NN method can be used to produce the actual count predictions as follows:

	count.pred																																		
test.response	-19	-18	-17	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	16	17	18
-32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-18	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	1	0	5	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-17	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-15	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	9	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-13	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	3	14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	6	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
-11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	8	0	2	0	1	0	0	2	0	0	0	0	0	0	0	0	0
-10	0	0	0	0	0	0	0	0	0	0	0	0	1	2	0	1	0	12	23	7	1	1	2	0	0	0	0	0	0	0	0	0	0	0	0
-9	0	0	0	0	0	0	0	0	0	0	0	0	1	4	0	6	3	7	18	4	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0
-8	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	7	5	24	33	7	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-7	0	0	0	0	0	0	0	0	0	0	0	0	4	4	3	17	7	21	60	7	4	1	0	0	0	0	0	0	0	0	0	0	0	0	0
-6	0	0	0	0	1	0	0	0	0	0	0	1	2	4	14	11	18	48	87	12	5	2	0	0	0	1	0	0	0	0	0	0	0	0	0
-5	0	0	1	0	1	0	0	0	0	0	1	3	0	5	13	22	44	80	163	38	21	7	1	3	2	1	0	1	0	0	0	0	0	0	0
-4	0	0	0	0	0	3	0	1	0	3	2	6	14	20	18	33	55	162	344	61	30	19	5	4	1	4	0	0	0	1	0	0	0	0	0
-3	0	0	0	0	0	3	0	4	1	1	2	0	10	21	23	57	102	307	696	117	46	13	9	6	4	1	1	3	1	0	0	0	0	0	0
-2	0	0	0	0	2	0	2	6	4	6	2	8	11	46	64	115	192	665	1592	300	110	41	19	11	6	6	2	1	0	1	0	0	0	0	0
-1	0	0	1	0	0	1	1	2	3	4	20	12	27	91	112	259	466	1580	5093	921	288	106	46	30	22	15	7	2	1	1	1	0	0	0	0
0	1	0	0	0	1	2	1	5	11	8	17	29	60	110	214	422	1068	3848	17519	2910	827	319	122	61	46	38	13	8	1	3	1	0	0	0	0
1	0	0	0	0	1	0	0	0	1	3	6	8	15	39	58	123	298	1074	4280	1124	317	178	65	30	35	27	2	4	3	4	4	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	3	4	4	9	21	47	116	311	1203	423	144	60	28	10	5	8	1	2	0	1	0	0	1	0	0
3	0	0	0	0	0	0	0	4	0	0	1	1	1	9	10	11	43	139	568	171	86	38	12	12	8	14	3	0	1	1	3	0	0	0	0
4	0	0	0	0	0	0	0	1	0	0	0	0	2	2	9	12	28	66	212	72	31	20	12	7	2	4	1	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	2	2	4	8	14	42	94	38	19	19	8	2	3	3	2	2	1	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	4	2	0	5	15	68	19	10	2	1	2	0	1	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	4	2	9	49	23	7	3	2	0	0	1	0	0	0	0	0	0	0	0	0