CSC478 Final Project - Analysis of Bike Sharing Data

Cecil Beeland
College of Computing & Digital Media, DePaul University, Chicago, IL, 60604

ABSTRACT
Bike sharing systems are new generation of traditional bike rentals where whole process from membership, rental and return back has become automatic. Through these systems, user is able to easily rent a bike from a particular position and return back at another position. Currently, there are about over 500 bike-sharing programs around the world which is composed of over 500 thousands bicycles. Today, there exists great interest in these systems due to their important role in traffic, environmental and health issues.

Apart from interesting real world applications of bike sharing systems, the characteristics of data being generated by these systems make them attractive for research. Opposed to other transport services such as bus or subway, the duration of travel, departure and arrival position is explicitly recorded in these systems. This feature turns bike sharing system into a virtual sensor network that can be used for sensing mobility in the city. Hence, it is expected that most of important events in the city could be detected via monitoring these data.

In this paper, I do preliminary analysis of a bike-share dataset and attempt to leverage the features in the dataset to predict the count of bike-share rentals for a given day, classify a given record by weather, and classify a given record by whether or not it is a working day.

1. INTRODUCTION

For this exercise, I focused exclusively on the following features of the dataset: Season (values 1-4), Year, Month, Hour, Working-day (1 if work day, 0 if weekend or holiday), Weather Situation (values 1-4, with 1 being clear weather and 4 being heavy inclement weather), Temperature, Humidity, Windspeed, and Bike-Share Rental Count.
A preliminary analysis of the dataset reveals some visible patterns in the data. *Figures 1 and 2* (below) highlight the differences between bike-share usage on a working day vs a non-working day. In these plots, you can see that on a working day, average hourly usage has higher maximums, with two peaks corresponding to the “rush hours” before and after working hours. Alternatively, on non-working days, there is a single normal distribution that peaks around early afternoon. There is also a small bump in usage at the very beginning of the day (hours 0-3) that corresponds with late-night usage from the previous evening.

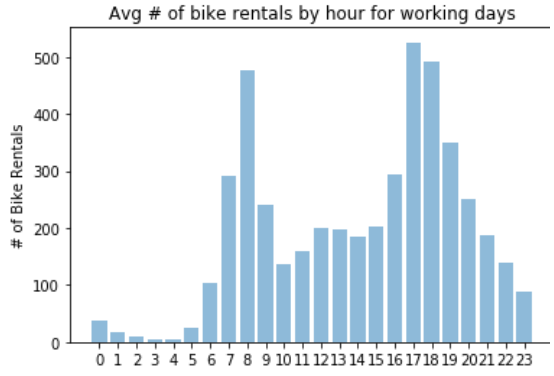

Figure 1: Average Number of Bike Rentals by Hour for Working Days

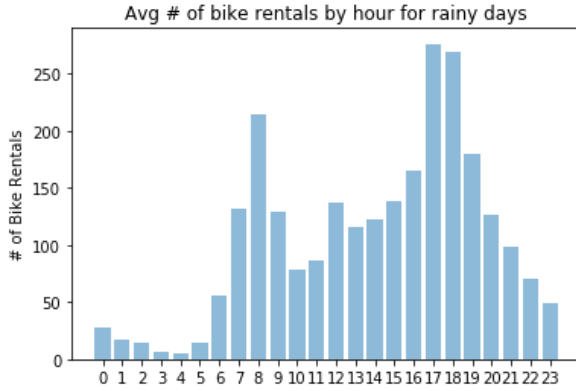

Figure 3: Average Number of Bike Rentals by Hour for Rainy Days

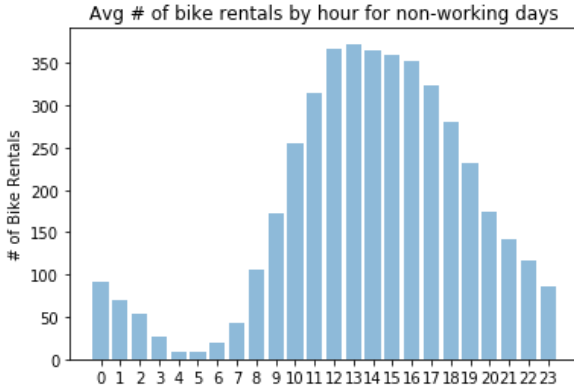

Figure 2: Average Number of Bike Rentals by Hour for Non-Working Days

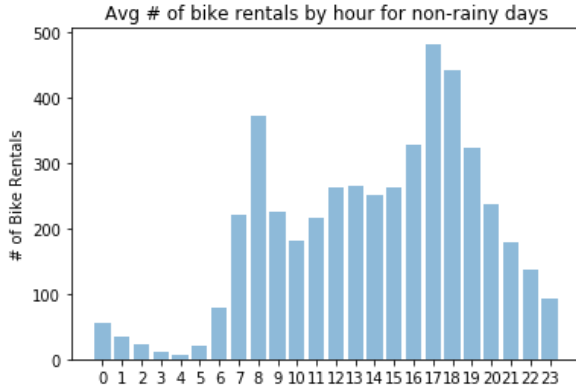

Figure 4: Average Number of Bike Rentals by Hour for Non-Rainy Days

Figures 3 and 4 (below) highlight the differences between bike-share usage on a rainy day vs a non-rainy day. In these plots, I am considering a day to be "rainy" if the Weather Situation feature is measured at 3 or higher (out of a range of 1-4). Here you can see that both rainy and non-rainy day usage have similar distributions with peaks around 8am and 5pm, but on a rainy day, average hourly usage has much lower maximums, with rainy days peaking around 250 rentals /

Additionally, I analyzed the dataset for outlier records. I plotted each feature on a histogram and looked for records that would stand apart from the general distribution. Additionally, I looked for records that had features with values that did not make sense for the data. Finally, I looked for records had features that were missing values. I identified no outliers throughout all of the outliers analysis, indicating that this dataset is much cleaner than most and has likely been pre-processed and cleaned for data-analysis.

hour, and non-rainy days peaking at nearly 500 rentals / hour.

2. PREDICTING BIKE-SHARE USAGE

I used a linear regression algorithm to attempt to predict the count of bike rentals for a given record in the dataset. The linear regression algorithm was fit against all of the features in the dataset except for the bike-rental count feature. Additionally, prior to execution of the prediction, the dataset was run through an Imputer step, imputing any missing values, as well as a Standard Scaler step, that standardize the features by removing the mean and scaling to unit variance.

To measure the effectiveness of the prediction, I used the Root Mean Squared Error (RMSE), which is a standard metric used to measure of the differences between values predicted by a model vs. the values actually observed.

The model was moderately effective, with an RMSE of 143.73, which means that on average the prediction was off by approximately 140 rentals. I expect that the accuracy of this model could be improved with further tuning of the model.

3. PREDICTING RAINY DAYS FROM BIKE-SHARE DATA

I used a K-Nearest Neighbors algorithm to attempt to classify the weather conditions for a given record in the dataset. The K-Nearest Neighbors algorithm was fit against all of the non-weather-related features in the dataset, to include: Season (values 1-4), Year, Month, Hour, Working-day (1 if work day, 0 if weekend or holiday), and Bike-Share Rental Count. Additionally, prior to execution of the prediction, the dataset was run through an Imputer step, imputing any missing values, as well as a Standard

To measure the effectiveness of the prediction, I generated a confusion matrix and used the components of the matrix (True Positive count, True Negative Count, False Positive Count, and False Negative Count) to calculate the overall accuracy, as well as the F1 Score, which is a standard metric used to measure the accuracy of a predicted classification. It considers both the precision p and the recall r of the test to compute the score.

The model was relatively poor at classifying the records. While the classifications had an overall accuracy of 91.98%, the F1-Score was a low 0.298, indicating that the classification is much more effective with one class than the other. I expect that the accuracy of this model could be improved with further tuning of the model, as well as improved sorting of the dataset into training and testing subsets so that both classes are equally represented.

[7845 117]
[580 148]

Figure 5: Rainy-Day Classification Confusion Matrix

4. PREDICTING WORKDAYS FROM BIKE-SHARE DATA

I used a K-Nearest Neighbors algorithm to attempt to classify whether a given record in the dataset represented a working day or a non-working day. The K-Nearest Neighbors algorithm was fit against all of the features in the dataset except for the workday feature. Additionally, prior to execution of the prediction, the dataset was run through an Imputer step, imputing any missing values, as well as a Standard Scaler step, that

Scaler step, that standardize the features by removing the mean and scaling to unit variance.
standardize the features by removing the mean and scaling to unit variance.

[7845 117]
[580 148]

Figure 5: Rainy-Day Classification Confusion Matrix

5. PREDICTING WORKDAYS FROM BIKE-SHARE DATA

I used a K-Nearest Neighbors algorithm to attempt to classify whether a given record in the dataset represented a working day or a non-working day. The K-Nearest Neighbors algorithm was fit against all of the features in the dataset except for the workday feature. Additionally, prior to execution of the prediction, the dataset was run through an Imputer step, imputing any missing values, as well as a Standard Scaler step, that standardize the features by removing the mean and scaling to unit variance.

To measure the effectiveness of the prediction, I generated a confusion matrix and used the components of the matrix (True Positive count, True Negative Count, False Positive Count, and False Negative Count) to calculate the overall accuracy, as well as the F1 Score.

The model was moderately effective, with an overall accuracy of 68.78% and the F1-Score at 0.815, indicating that the classification is fairly consistent across both classes, but is only moderately effective, with an accuracy rate in the high 60s. I expect that the accuracy of this model could be improved with further tuning of the model, as well as improved sorting

[1 2713]
[0 5976]

Figure 6: Workday Classification Confusion Matrix

6. PREDICTING RAINY DAYS FROM UNLABELED BIKE-SHARE DATA

I used a K-Means algorithm to cluster records of the dataset into two clusters. The K-Means algorithm was fit against all of the non-weather-related features in the dataset, to include: Season (values 1-4), Year, Month, Hour, Working-day (1 if work day, 0 if weekend or holiday), and Bike-Share Rental Count. Additionally, prior to execution of the prediction, the dataset was run through an Imputer step, imputing any missing values, as well as a Standard Scaler step, that standardize the features by removing the mean and scaling to unit variance.

To measure the effectiveness of the prediction, I generated a confusion matrix and used the components of the matrix (True Positive count, True Negative Count, False Positive Count, and False Negative Count) to calculate the overall accuracy, as well as the F1 Score.

The model was relatively poor at clustering the records in such a way that would allow us to classify one cluster as rainy-day records and the other with non-rainy-day records. The resulting clusters had an overall accuracy of 53.3% and the F1-Score at 0.156, indicating that the classification is much more with one class than the other, and generally poor for bot. I expect that the accuracy of this model could be improved with further tuning of the model, as well as improved sorting of the dataset into training and testing

of the dataset into training and testing subsets so that both classes are equally represented.

subsets so that both classes are equally represented.

[4256 3706]
[352 376]

Figure 7: Rainy Day Classification (using K-Means clustering) Confusion Matrix

APPENDIX

Python Code:

```
import pandas as pd
import numpy as np
import random
import re

from math import log

import sklearn
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler, Imputer
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression, LinearRegression
from sklearn.neighbors import KNeighborsRegressor
from sklearn.cluster import KMeans
from sklearn.metrics import confusion_matrix
from sklearn.metrics import mean_squared_error

import matplotlib
import matplotlib.pylab as plt
%matplotlib inline

df = pd.read_csv('hour.csv')
```

```
# Plot Avg bike rentals for working days
df = df[['season', 'yr', 'mnth', 'hr', 'workingday', 'weathersit', 'atemp', 'hum', 'windspeed',
'cnt']].copy()

objects = np.arange(24)
y_pos = np.arange(24)

avgHourlyWorkDayRentals = df[['hr','cnt']][(df['workingday'] == 1)].groupby(['hr']).mean()

plt.bar(y_pos, avgHourlyWorkDayRentals['cnt'], align='center', alpha=0.5)
plt.xticks(y_pos, objects)
plt.ylabel('# of Bike Rentals')
plt.title('Avg # of bike rentals by hour for working days')

# Plot Avg bike rentals for non-working days
avgNonWorkDayHourlyRentals = df[['hr','cnt']][(df['workingday'] ==
0)].groupby(['hr']).mean()

plt.bar(y_pos, avgNonWorkDayHourlyRentals['cnt'], align='center', alpha=0.5)
plt.xticks(y_pos, objects)
plt.ylabel('# of Bike Rentals')
plt.title('Avg # of bike rentals by hour for non-working days')

# Plot Avg bike rentals for rainy days
avgRainyDayHourlyRentals = df[['hr','cnt']][(df['weathersit'] >= 3)].groupby(['hr']).mean()

plt.bar(y_pos, avgRainyDayHourlyRentals['cnt'], align='center', alpha=0.5)
plt.xticks(y_pos, objects)
plt.ylabel('# of Bike Rentals')
plt.title('Avg # of bike rentals by hour for rainy days')

# Plot Avg bike rentals for non-rainy days
avgNonRainyDayHourlyRentals = df[['hr','cnt']][(df['weathersit'] <=
2)].groupby(['hr']).mean()

plt.bar(y_pos, avgNonRainyDayHourlyRentals['cnt'], align='center', alpha=0.5)
plt.xticks(y_pos, objects)
plt.ylabel('# of Bike Rentals')
plt.title('Avg # of bike rentals by hour for non-rainy days')

# Build Linear Regression Pipeline
linear_reg_pipeline = Pipeline([
   ('my_imputer', Imputer()),
   ('my_std_scaler', StandardScaler()),
   ('masteralg', LinearRegression())
])
```

```
df = df.sample(frac=1)

#### Predict bike-rental count from features
y_col = 'cnt'
n = (len(df)/2)

df_train, df_test = df.iloc[:n].copy(), df.iloc[n:].copy()
print len(df_train), len(df_test)

X_train, y_train = df_train[df_train.columns.drop(y_col)], df_train[y_col]
X_test, y_test = df_test[df_test.columns.drop(y_col)], df_test[y_col]

linear_reg_pipeline.fit(X_train, y_train)
y_pred = linear_reg_pipeline.predict(X_test)

RMSE = (((y_test - y_pred)**2).sum()/len(y_test))**0.5
print RMSE

#### Weather Prediction with non-weather
df_weather_predict = df.copy()

# Define boolean feature for whether or not it is rainy (weathersit > 2)
df_weather_predict['isRainy'] = 0
df_weather_predict.loc[(df.weathersit>2),'isRainy'] = 1

#Drop non weather columns
df_weather_predict =
df_weather_predict[['season','yr','mnth','hr','workingday','cnt','isRainy']]

y_col = 'isRainy'
n = (len(df_weather_predict)/2)

df_train, df_test = df_weather_predict.iloc[:n].copy(), df_weather_predict.iloc[n:].copy()

X_train, y_train = df_train[df_train.columns.drop(y_col)], df_train[y_col]
X_test, y_test = df_test[df_test.columns.drop(y_col)], df_test[y_col]

logistic_reg_pipeline = Pipeline([
   ('my_imputer', Imputer()),
   ('my_std_scaler', StandardScaler()),
   ('masteralg', KNeighborsClassifier(n_neighbors=3))
])

logistic_reg_pipeline.fit(X_train, y_train)
y_pred = logistic_reg_pipeline.predict(X_test)

print 'Confusion Matrix:'
```

```
print(sklearn.metrics.confusion_matrix(y_test, y_pred))
tn, fp, fn, tp = sklearn.metrics.confusion_matrix(y_test, y_pred).ravel()

num_predicted_correctly = tn + tp
tpr = float(tp)/(tp+fn) # precision / true positive rate
tnr = float(tn)/(tn+fp) # true negative rate
ppv = float(tp)/(tp+fp) # recall / positive predictive value
npv = float(tn)/(tn+fn) # negative predictive value

f1 = (2.0/(1.0/ppv+1.0/tpr)) #sklearn.metrics.f1_score(y_test, y_pred)
print '% Correctly Predicted: ' + str(float(num_predicted_correctly)/len(X_test))
print 'F1 Score: ' + str(f1)

#### Workingday prediction with non-workingday features
df_workingday_predict = df.copy()

y_col = 'workingday'
n = (len(df_workingday_predict)/2)

df_train, df_test = df_workingday_predict.iloc[:n].copy(),
df_workingday_predict.iloc[n:].copy()

X_train, y_train = df_train[df_train.columns.drop(y_col)], df_train[y_col]
X_test, y_test = df_test[df_test.columns.drop(y_col)], df_test[y_col]

k_neighbors_pipeline = Pipeline([
   ('my_imputer', Imputer()),
   ('my_std_scaler', StandardScaler()),
   ('masteralg', KNeighborsClassifier(n_neighbors=3))
])

k_neighbors_pipeline.fit(X_train, y_train)
y_pred = k_neighbors_pipeline.predict(X_test)

print 'Confusion Matrix:'
print(sklearn.metrics.confusion_matrix(y_test, y_pred))
tn, fp, fn, tp = sklearn.metrics.confusion_matrix(y_test, y_pred).ravel()

num_predicted_correctly = tn + tp
tpr = float(tp)/(tp+fn) # precision / true positive rate
tnr = float(tn)/(tn+fp) # true negative rate
ppv = float(tp)/(tp+fp) # recall / positive predictive value
npv = float(tn)/(tn+fn) # negative predictive value

f1 = (2.0/(1.0/ppv+1.0/tpr)) #sklearn.metrics.f1_score(y_test, y_pred)
print '% Correctly Predicted: ' + str(float(num_predicted_correctly)/len(X_test))
print 'F1 Score: ' + str(f1)
```

```
#### K-Means Clustering (2-clusters) predicting weather with non-weather features
df_weather_predict = df.copy()

# Define boolean feature for whether or not it is rainy (weathersit > 2)
df_weather_predict['isRainy'] = 0
df_weather_predict.loc[(df.weathersit>2),'isRainy'] = 1

#Drop non weather columns
df_weather_predict =
df_weather_predict[['season','yr','mnth','hr','workingday','cnt','isRainy']]

y_col = 'isRainy'
n = (len(df_weather_predict)/2)

df_train, df_test = df_weather_predict.iloc[:n].copy(), df_weather_predict.iloc[n:].copy()

X_train, y_train = df_train[df_train.columns.drop(y_col)], df_train[y_col]
X_test, y_test = df_test[df_test.columns.drop(y_col)], df_test[y_col]

logistic_reg_pipeline = Pipeline([
   ('my_imputer', Imputer()),
   ('my_std_scaler', StandardScaler()),
   ('masteralg', KMeans(n_clusters=2))
])

logistic_reg_pipeline.fit(X_train, y_train)
y_pred = logistic_reg_pipeline.predict(X_test)

print 'Confusion Matrix:'
print(sklearn.metrics.confusion_matrix(y_test, y_pred))
tn, fp, fn, tp = sklearn.metrics.confusion_matrix(y_test, y_pred).ravel()

num_predicted_correctly = tn + tp
tpr = float(tp)/(tp+fn) # precision / true positive rate
tnr = float(tn)/(tn+fp) # true negative rate
ppv = float(tp)/(tp+fp) # recall / positive predictive value
npv = float(tn)/(tn+fn) # negative predictive value

f1 = (2.0/(1.0/ppv+1.0/tpr)) #sklearn.metrics.f1_score(y_test, y_pred)
print '% Correctly Predicted: ' + str(float(num_predicted_correctly)/len(X_test))
print 'F1 Score: ' + str(f1)
```