



Capstone Project : Capital Bikeshare Rental Prediction

Presentation for IBM Specialization in
Advanced Data Science via Coursera



Hemant, K. //  khemanta // 
April 1, 2020

 attendee @ Coursera | IBM Advance Data Science Specialization



Grading Guidelines: For Video Presentation

Please watch the video and assign points as described below. For each item you find coverage in the video please assign one point.

- 1/a. Use Case
- 1/b. Data Set
- 1/c. Data Quality Assessment
- 2/a. Data Exploration (e.g. correlation between columns)
- 2/b. Data Visualization (e.g. value distribution of columns)
- 2/c. At least one Feature Engineering (e.g. imputing missing values) applied
- 3/a. Selection and justification of Model Performance Indicator (e.g. F1 score)
- 3/b. At least one traditional Machine Learning Algorithm and one DeepLearning Algorithm applied and demonstrated
- 3/c. Model performance between different feature engineerings and models compared and documented



1. Business Problem, Use Case, and Data Understanding
2. Data Wrangling: EDA, Visualization, and ETL
3. Modeling Techniques: Machine Learning & Deep Learning
 - 3.a. Machine Learning : Random Forest
 - 3.b. Deep Learning : Neural Network with Back-Prop



1. Business Problem, Use Case, and Data Understanding



Business Problems

1. Capital Bikeshare, a bike sharing app, want to understand - The rental demand in Washington DC area based on historical data for better placing the bikes at designated places as per demands.
2. Also, they want to understand - Are there emerging patterns in rental due to changing of seasons, weather patterns and even time?

Get started on this competition through [Kaggle Scripts](#)

Bike sharing systems are a means of renting bicycles where the process of obtaining membership, rental, and bike return is automated via a network of kiosk locations throughout a city. Using these systems, people are able rent a bike from a one location and return it to a different place on an as-needed basis. Currently, there are over 500 bike-sharing programs around the world.

The data generated by these systems makes them attractive for researchers because the duration of travel, departure location, arrival location, and time elapsed is explicitly recorded. Bike sharing systems therefore function as a sensor network, which can be used for studying mobility in a city. 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.



Acknowledgements

Kaggle is hosting this competition for the machine learning community to use for fun and practice. This dataset was provided by Hadi Fanaee Tork using data from [Capital Bikeshare](#). We also thank the UCI machine learning repository for [hosting the dataset](#). If you use the problem in publication, please cite:

Fanaee-T, Hadi, and Gama, Joao, *Event labeling combining ensemble detectors and background knowledge*, Progress in Artificial Intelligence (2013): pp. 1-15, Springer Berlin Heidelberg.

capital bikeshare

[How It Works](#) [Pricing](#) [System Map](#) [Explore](#) [Help](#)

System Data

Where do Capital Bikeshare riders go? When do they ride? How far do they go? Which stations are most popular? What days of the week are most rides taken on? We've heard all of these questions since we launched in 2010, and we're glad to provide the data that shows you the answers from our first trip to today.

We invite developers, engineers, statisticians, artists, academics and other interested members of the public to use the data we provide for analysis, development, visualization or whatever else moves you.

This data is provided according to the [Capital Bikeshare Data License Agreement](#).

<https://www.kaggle.com/c/bike-sharing-demand>

<https://www.capitalbikeshare.com/system-data>

[Get started on this competition through Kaggle Scripts](#)

Bike sharing systems are a means of renting bicycles where the process of obtaining membership, rental, and bike return is automated via a network of kiosk locations throughout a city. Using these systems, people are able rent a bike from a one location and return it to a different place on an as-needed basis. Currently, there are over 500 bike-sharing programs around the world.

The data generated by these systems makes them attractive for researchers because the duration of travel, departure location, arrival location, and time elapsed is explicitly recorded. Bike sharing systems therefore function as a sensor network, which can be used for studying mobility in a city. 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.



Acknowledgements

Kaggle is hosting this competition for the machine learning community to use for fun and practice. This dataset was provided by Hadi Fanaee Tork using data from [Capital Bikeshare](#). We also thank the UCI machine learning repository for [hosting the dataset](#). If you use the problem in publication, please cite:

Fanaee-T, Hadi, and Gama, Joao, *Event labeling combining ensemble detectors and background knowledge*, Progress in Artificial Intelligence (2013): pp. 1-15, Springer Berlin Heidelberg.

- Year 2011-2012 Data Taken
- Washington D.C. area central metro station
- hourly and daily data from Sep 2011 to Aug 2012
 - hour.csv : bike sharing counts aggregated on hourly basis.
Records: 17379 hours
 - day.csv : bike sharing counts aggregated on daily basis.
Records: 731 days

<https://www.kaggle.com/c/bike-sharing-demand>



<https://github.com/khemanta/IBM-Advanced-Data-Science-Capstone/tree/master/assets/data>

```
=====
Files
=====
```

- Readme.txt
- hour.csv : bike sharing counts aggregated on hourly basis. Records: 17379 hours
- day.csv - bike sharing counts aggregated on daily basis. Records: 731 days

```
=====
Dataset characteristics
=====
```

Both hour.csv and day.csv have the following fields, except hr which is not available in day.csv

- instant: record index
- dteday : date
- season : season (1:springer, 2:summer, 3:fall, 4:winter)
- yr : year (0: 2011, 1:2012)
- mnth : month (1 to 12)
- hr : hour (0 to 23)
- holiday : weather day is holiday or not (extracted from <http://dchr.dc.gov/page/holiday-schedule>)
- weekday : day of the week
- workingday : if day is neither weekend nor holiday is 1, otherwise is 0.
- + weathersit :
 - 1: Clear, Few clouds, Partly cloudy, Partly cloudy
 - 2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
 - 3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds
 - 4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog
- temp : Normalized temperature in Celsius. The values are divided to 41 (max)
- atemp: Normalized feeling temperature in Celsius. The values are divided to 50 (max)
- hum: Normalized humidity. The values are divided to 100 (max)
- windspeed: Normalized wind speed. The values are divided to 67 (max)
- casual: count of casual users
- registered: count of registered users
- cnt: count of total rental bikes including both casual and registered

The quality of the data

Observations: The data quality is good and there is no missing value. However, there are extreme values and would require scaling which we will discover in Exploratory Data Analysis steps and Feature Engineering steps.

<https://github.com/khemanta/IBM-Advanced-Data-Science-Capstone/blob/master/Capital-Bikeshare-Rental-Demand-Forecast-1-Random-Forest.ipynb>

```
In [5]: #Get the data statistics for each column
hours_data.describe()
```

	season	month	hour	holiday	weekday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
count	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000
mean	2.501640	6.537775	11.546752	0.028770	3.003683	0.682721	1.425283	0.496987	0.475775	0.627229	0.190098	35.676218	153.786869	189.463088
std	1.106918	3.438776	6.914405	0.167165	2.005771	0.465431	0.639357	0.192556	0.171850	0.192930	0.122340	49.305030	151.357286	181.387599
min	1.000000	1.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.020000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000
25%	2.000000	4.000000	6.000000	0.000000	1.000000	0.000000	1.000000	0.340000	0.333300	0.480000	0.104500	4.000000	34.000000	40.000000
50%	3.000000	7.000000	12.000000	0.000000	3.000000	1.000000	1.000000	0.500000	0.484800	0.630000	0.194000	17.000000	115.000000	142.000000
75%	3.000000	10.000000	18.000000	0.000000	5.000000	1.000000	2.000000	0.660000	0.621200	0.780000	0.253700	48.000000	220.000000	281.000000
max	4.000000	12.000000	23.000000	1.000000	6.000000	1.000000	4.000000	1.000000	1.000000	1.000000	0.850700	367.000000	886.000000	977.000000

2. Data Wrangling: EDA, Visualization, and ETL



Data attributes and types transformation: Numeric to Categorical

Converting the few the data attributes as numeric values to relevant useful categorical type.

```
In [6]: # Converting the columns type to category
hours_data['datetime'] = hours_data.datetime.astype('category')
hours_data['season'] = hours_data.season.astype('category')
hours_data['month'] = hours_data.month.astype('category')
hours_data['hour'] = hours_data.hour.astype('category')
hours_data['holiday'] = hours_data.holiday.astype('category')
hours_data['weekday'] = hours_data.weekday.astype('category')
hours_data['workingday'] = hours_data.workingday.astype('category')
hours_data['weather'] = hours_data.weather.astype('category')

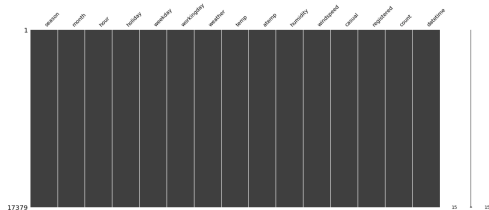
hours_data.dtypes
```

```
Out[6]: season      category
        month      category
        hour       category
        holiday    category
        weekday    category
        workingday category
        weather    category
        temp       float64
        atemp      float64
        humidity   float64
        windspeed  float64
        casual     int64
        registered int64
        count      int64
        datetime   category
        dtype: object
```

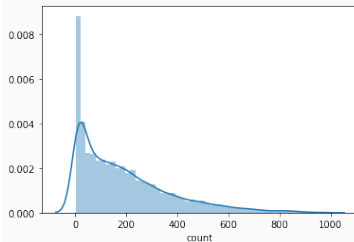
Missing Value and Treatment

```
In [7]: # Checking for any NULL value in the data  
ms.matrix(hours_data)
```

```
Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0x12e8eb110>
```



(a.) Missing values in the total data per attribute wise



(b.) The overall distribution of the data

No missing values, as data is rich and cleaned already when provided.

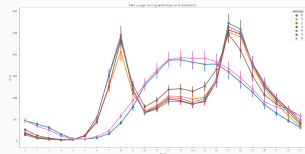
All the 15 attributes are already imputed.

Visualization of the Data: Descriptive Data Analysis

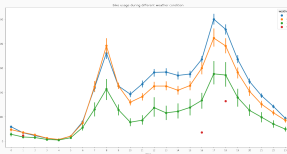


Bike rentals during different seasons and weather conditions : averaged over Weekdays vs. Weekends, Seasons, and Weathers □ 0: Sunday, 1: Monday, 2: Tuesday, 3: Wednesday, 4: Thursday, 5: Friday, 6: Saturday;

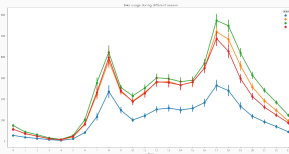
■ 1: sunny/clear , 2: cloudy/misty, 3: foggy/snowy-light , 4: rainy/snowy-heavily ■ 1: Spring, 2: Summer, 3: Fall, 4: Winter □



(a.) Rental over Weekdays/Weekend (0-6)

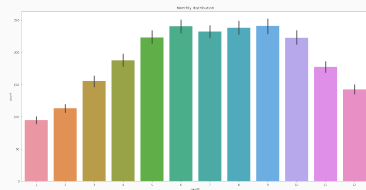


(b.) Rental in 4 Seasons (1-4), and

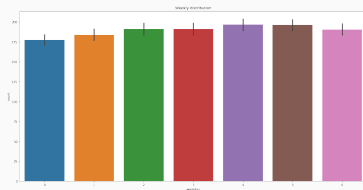


(c.) Different Weathers Conditions (1-4)

Bike rental monthly and weekly conditions over a year

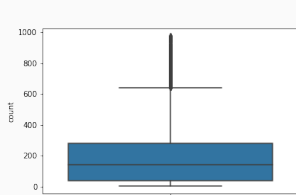


d.) Monthly pattern of bike rental : Jan-Dec (1-12) months;

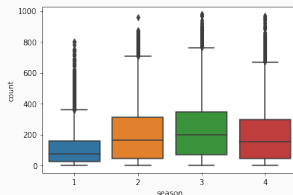


e.) Weekly pattern of bike rental : Sunday to Saturday (0-6)

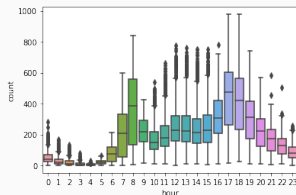
1. Rental data distribution across time (hourly) and seasonal effect on rental demand over the years



(a.) Boxplot of bike rental data distribution

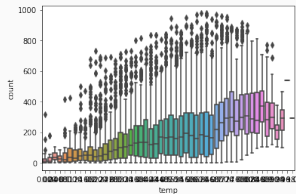


(b.) Rental distribution over Seasons (1-4)

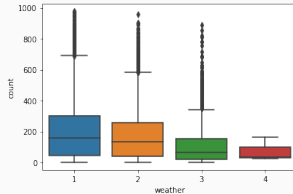


(c.) Rental distribution hourly (0-23) hrs

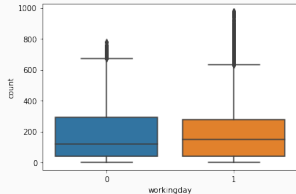
2. Bike rental distribution over different weather and climatic conditions (temperature, humidity, etc.), and holidays



(d.) Bike Rental vs Temperature distribution



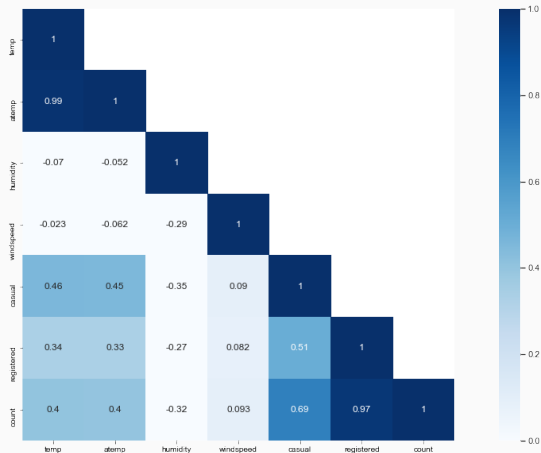
(e.) Distribution over Weather (1-4), and



(f.) Working day vs. Holiday (0-1)

Inference & Observation :

The correlation matrix shows high positive correlation between **temp** and **atemp** so we can remove **atemp** column. The **casual** and **registered** data contains information about the **count** and it could lead to data leakage, hence we can remove both of them.





1. Dropped highly collinear attributes and linearly dependent to each other, thus reduced 15 to 12.

```
In [29]: hours_data.shape
Out[29]: (17379, 15)

In [30]: # Removing highly correlated features
hours_data = hours_data.drop(['atemp', 'casual', 'registered'], axis = 1)
hours_data.shape
Out[30]: (17379, 12)
```

2. Feature engineering using One hot encoding for categorical attributes

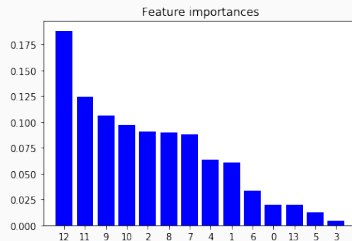
```
In [31]: #One hot encoding
data_dummy = hours_data

def generate_dummies(df, dummy_column):
    dummies = pd.get_dummies(df[dummy_column], prefix=dummy_column)
    df = pd.concat([df, dummies], axis=1)
    return df

dummy_data = pd.DataFrame.copy(hours_data)
dummy_columns = ["season", "month", "hour", "holiday", "weekday", "workingday", "weather"]
for dummy_column in dummy_columns:
    dummy_data = generate_dummies(dummy_data, dummy_column)

In [32]: # Removing the original columns
for dummy_column in dummy_columns:
    del dummy_data[dummy_column]

In [33]: dummy_data.shape
Out[33]: (17379, 60)
```



The relative importance of the features after feature engineering.

3. Modeling Techniques: Machine Learning & Deep Learning



Random Fores Model | Parameters:

```
regressor = RandomForestRegressor(n_estimators = 350, max_features = 'auto')
regressor.fit(X_train,y_train)

# Predicting the values
y_pred = regressor.predict(X_test)
y_pred = y_pred.astype(int)
```

Hyper-Parameters :

- a.) # of DecisionTree, i.e. $n_{estimators}$ = 350
- b.) Cross Validation, cv = 10
- c.) Cros Validation Sample with replacement, i.e., Bootstraped = True (default)

Results :

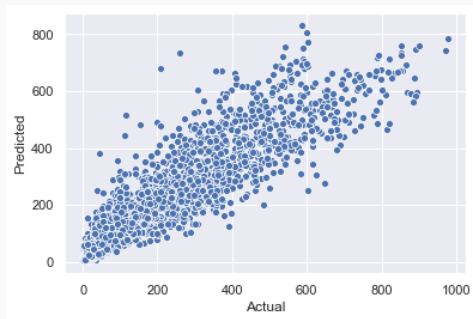
1.	Mean Accuracy	0.8361983320055169
2.	Mean Absolute Error	46.53279631760645
3.	Root Mean Squared Log Error (RSMLE)	0.457739960241472

The Machine Learning Model Random Forest Accuracy Metric.

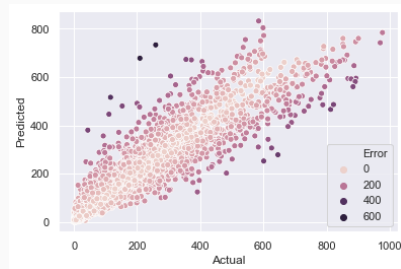
1	Mean Accuracy	83.62%
2	Mean Absolute Error	46.53
3	Root Mean Squared Log Error (RSMLE)	0.4577

Sample Prediction and Scatter Plot of Actual vs. Predicted

	Actual	Predicted	Error
0	425	378	47
1	88	105	17
2	4	12	8
3	526	421	105
4	13	25	12

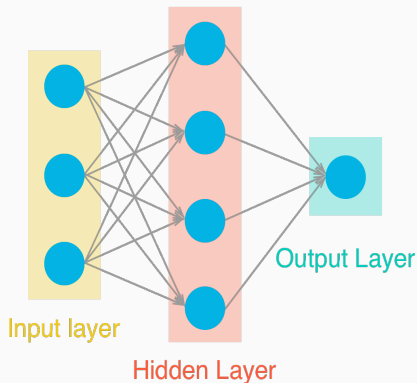


- The results seems to be good for Random Forest implementation.
- Accuracy rate is 83.62% and Mean Error is 45 which is reasonable in the sense of bike demand.
- The scatter plot with error as a metric of very few large errors are there.
- Mostly result of predicted vs actual is linear obvious from the scatter plot.
- Accuracy can be improvised with looking at other factors and feature engineering like seasonality, i.e. different model for different seasons could be one option worth trying, as exploratory data analysis suggest seasonal and weather variations.
- However, one aspect of the data is not looked at, i.e. place and nearby places. The geo-code availability could make possible the predictive power of clustered zones with spatial-and-temporal aspect for better prediction of shared bike.
- Another data is telecoms, to be factored in for people footfall and density may translate into better prediction. However this data is not available.
- With same data we can use some neural network techniques.



This will be our next step ... Deep Learning and Neural Network !

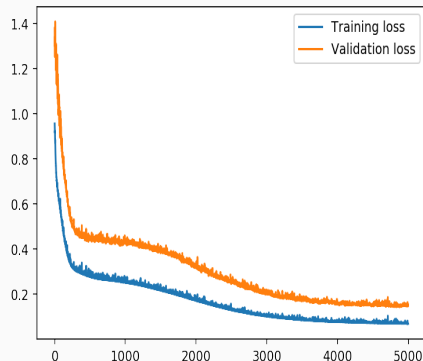
Neural Network Architecture



Setting the Hyper-Parameters :

- a.) # of Iterations = 5000
- b.) Learning Rate = 0.4

The Loss Function



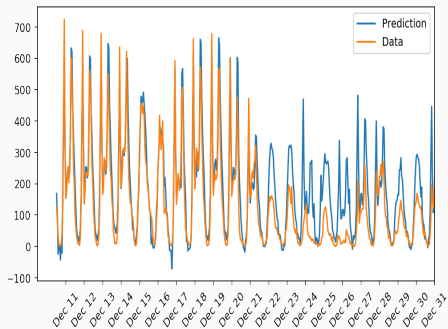
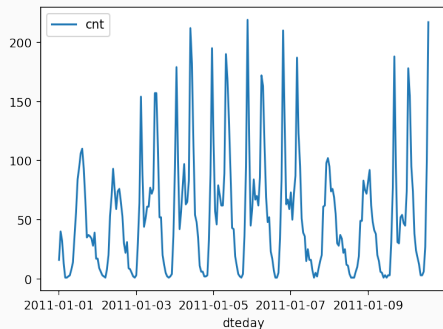
Setting the Hyper-Parameters :

- c.) # of Hidden Nodes = 10
- d.) Output Nodes = 1

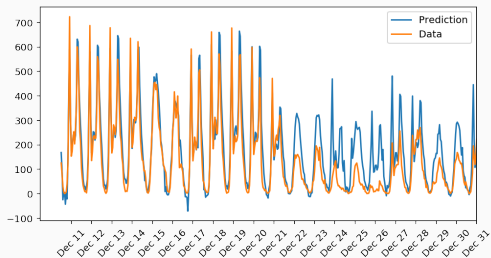
Neural Network Model Performance : Training vs. Validation

1	Training loss	0.068
2	Validation loss	0.152

Sample Predicted Results: Actual vs. Predicted



- The model gives a fairly good prediction for a normal day. It is off for days with special events, bad weather and holidays.
- We should divide training data into normal days and special days and build different NN for each type of days and use corresponding models to predict the data.
- Filtering data to remove outliers and data augmentation will improve the accuracy of the model.
- Adding more layers might improve the accuracy as more layers will help Neural Network to identify features at each layers and help to improve accuracy, i.e., explore some deep or recurrent network architecture.



This will be our next step ... **Recurrent Neural Network / LSTM / DeepNet/ etc. !**

Thank You !

for your patience 😊



hemant, k.

 khemanta **in** khemant

URL of this presentation: <https://github.com/khemanta/IBM-Advanced-Data-Science-Capstone/Capstone-Project-Presentation.pdf>