Data Analytics with Python

Certification Project

Firoz Syed



**Bike-Sharing Demand Analysis**

**Objective:**  Use data to understand what factors affect the number of bike trips. Make a predictive model to predict the number of trips in a particular hour slot, depending on the environmental conditions.

**Problem Statement:**

Lyft, Inc. is a transportation network company based in San Francisco, California and operating in 640 cities in the United States and 9 cities in Canada. It develops, markets, and operates the Lyft mobile app, offering car rides, scooters, and a bicycle-sharing system. It is the second largest rideshare company in the world, second to only Uber.

Lyft’s bike-sharing service is also among the largest in the USA. Being able to anticipate demand is extremely important for planning of bicycles, stations, and the personnel required to maintain these. This demand is sensitive to a lot of factors like season, humidity, rain, weekdays, holidays, and more. To enable this planning, Lyft needs to rightly predict the demand according to these factors.

**Domain:** General

**Analysis to be done:** Rightly predict the bike demand

**Content:** Dataset: Lyft bike-sharing data (hour.csv)

Fields in the data:

**- instant:** record index

**- dteday:** date

**- season:** season (1:spring, 2:summer, 3:fall, 4:winter)

**- yr: year** (0: 2011, 1: 2012)

**- mnth:** month (1 to 12)

**- hr:** hour (0 to 23)

**- holiday :** whether the day is a holiday or not

**- weekday :** day of the week

**- workingday :** if the day is neither weekend nor a holiday is 1, otherwise is 0

**- weathersit :**

- 1: Clear, Few clouds, Partly cloudy

- 2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist

- 3: Light Snow, Light Rain + Thunderstorm + Scattered clouds

- 4: Heavy Rain + Ice Pellets + Thunderstorm + Mist, Snow + Fog

**- temp :** normalized temperature in Celsius; the values are divided to 41 (max)

**- atemp:** normalized temperature felt 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

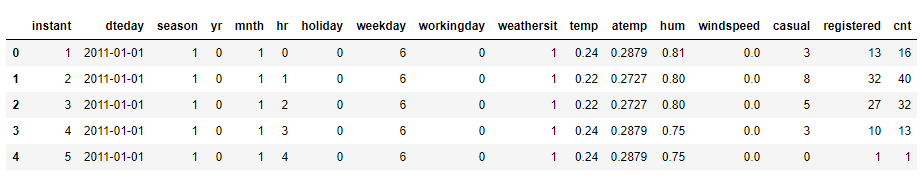
**- cnt:** count of total rental bikes including both casual and registered

**Steps to perform:**

As the first step, look at the null values in the file. A sanity check, to ensure that you have clean records and the data is good to go ahead, is very important. Then, you’ll do univariate and bivariate analyses to identify the patterns in the data and the nature of the individual features. This is a very important step as this helps to not only identify features which could be interesting for the predictive model later, but also helps understand what’s going on in the data. The EDA will help identify the need to apply transformations on the features before building the model. Finally, you will make a predictive model using linear regression.

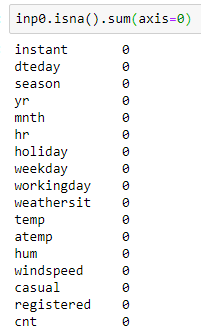
**Solution**

1. **Load the data file**

inp0 = pd.read\_csv("hour.csv")  
inp0.head()

1. **Check for null values in the data, drop records with NAs**

inp0.isna().sum(axis=0)



Looks like there are no records with null values. Looks good so far.

1. **Sanity checks:**
   1. **Check if registered + casual = cnt for all the records. The two must add to cnt, if not the row is junk and should be dropped.**

np.sum((inp0.casual + inp0.registered) != inp0.cnt)



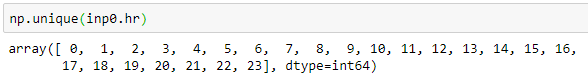
* 1. **Month values should be 1-12 only**

np.unique(inp0.mnth)

****

* 1. **Hour should be 0-23**

np.unique(inp0.hr)

****

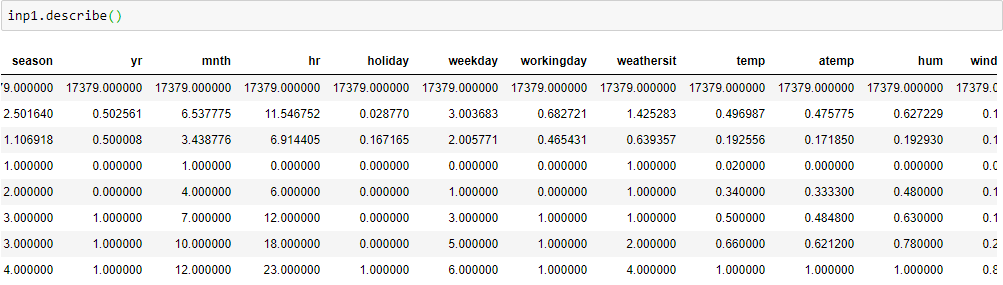
1. **Variables ‘casual’, ‘registered’ are redundant and need to be dropped. ‘Instant’ is the index, and needs to be dropped too. The date column dteday will not be used in the model building, and hence needs to be dropped. Create new dataframe named ‘inp1’.**

cols\_to\_drop = ['casual', 'registered', 'dteday', 'instant']

inp1 = inp0.drop(cols\_to\_drop, axis=1).copy()

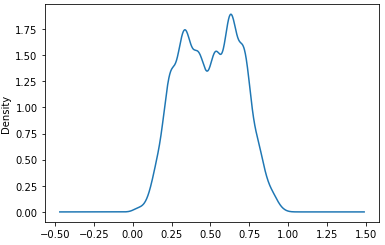
**5. Univariate analysis –**

* **Describe the numerical fields in the dataset using pandas describe method**

inp1.describe()

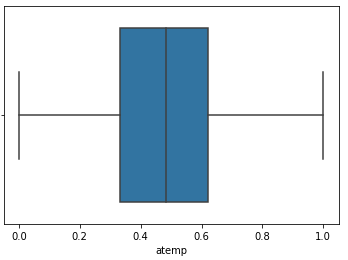
* **Make density plot for temp. This would give a sense of the centrality and the spread of the distribution.**

inp1.temp.plot.density()

****

* **Boxplot for atemp.** 
  + ***Are there any outliers?***

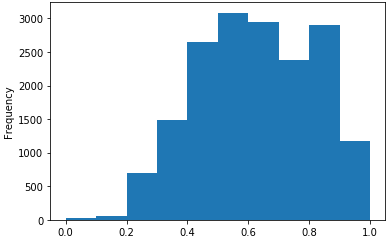
sns.boxplot(inp1.atemp)



There don’t seem to be any outliers for atemp.

* **Histogram for hum**
  + ***Do you detect any abnormally high values?***

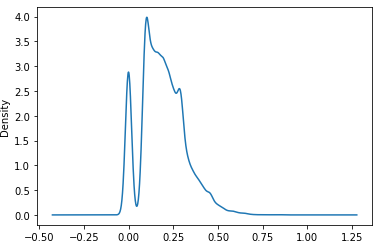
inp1.hum.plot.hist()



No visible abnormally high values

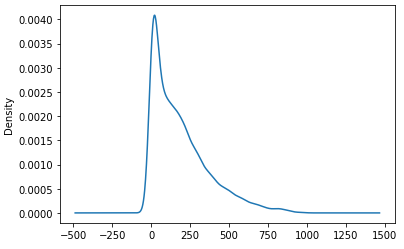
* **Density plot for windspeed**

inp1.windspeed.plot.density()

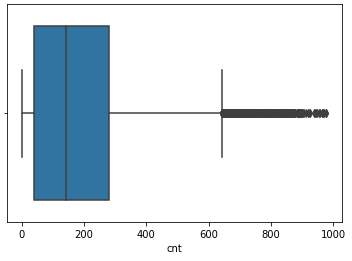
****

* **Box and density plot for cnt – this is the variable of interest.** 
  + ***Do you see any outliers in the boxplot?***
  + ***Does the density plot provide a similar insight?***

inp1.cnt.plot.density()

****

sns.boxplot(inp1.cnt)

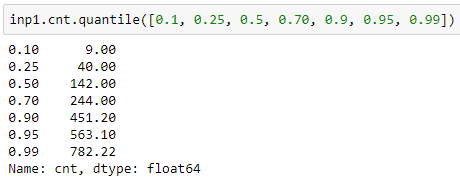
****

Both plots show similar picture – some high values are present in cnt.

**6. Outlier treatment –**

1. **Cnt – looks like some hours have rather high values of cnt. We’ll need to treat these outliers so that they don’t skew our analysis and our model.** 
   1. **Find out the following percentiles - 10, 25, 50, 75, 90, 95, 99**
   2. **Decide the cutoff percentile and drop records with values higher that the cutoff. Name the new dataframe ‘inp2’.**

inp1.cnt.quantile([0.1, 0.25, 0.5, 0.70, 0.9, 0.95, 0.99])

****

563 is the 95th percentile – only 5% records have a value higher than this. Taking this as the cutoff.

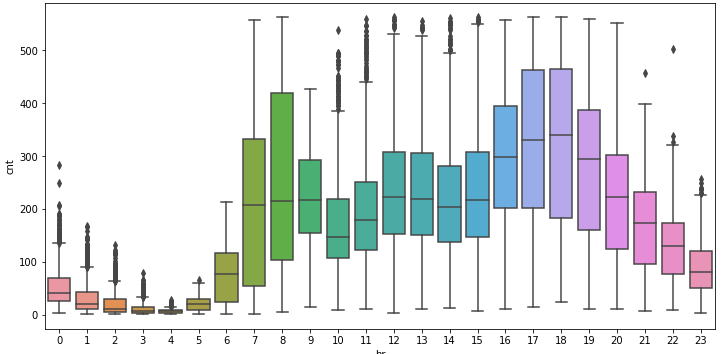
inp2 = inp1[inp1.cnt < 563].copy()

**7. Bi-variate analysis**

1. **Make box plot for cnt vs hr**
   1. ***What kind of pattern do you see?***

plt.figure(figsize=[12,6])

sns.boxplot("hr", "cnt", data=inp2)

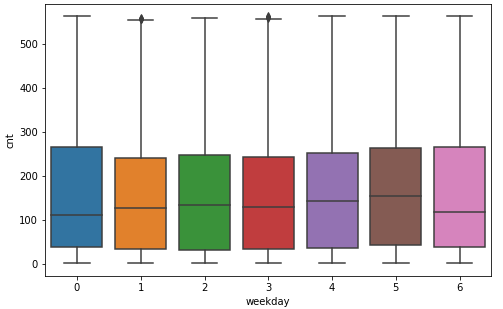


It’s evident that the peak hours are 5PM – 7PM, the hours 7-8AM also have high upper quartile. A hypothesis could be that a lot of people use the bikes for commute to workplace and back.

1. **Make boxplot for cnt vs weekday**
   1. ***Is there any difference in the rides by days of the week?***

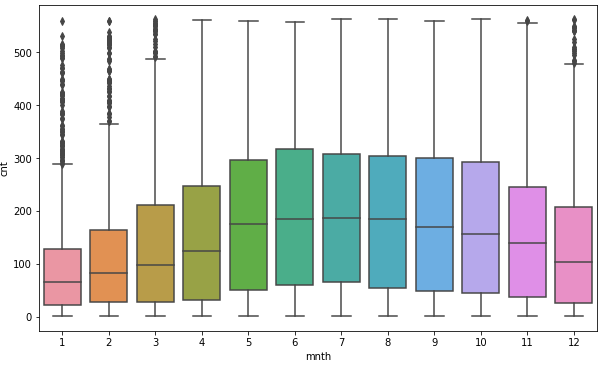
plt.figure(figsize=[8,5])

sns.boxplot("weekday", "cnt", data=inp2)

****

1. **Make boxplot for cnt vs month**
   1. ***Look at the median values. Any month(s) that stand out?***

plt.figure(figsize=[10,6])

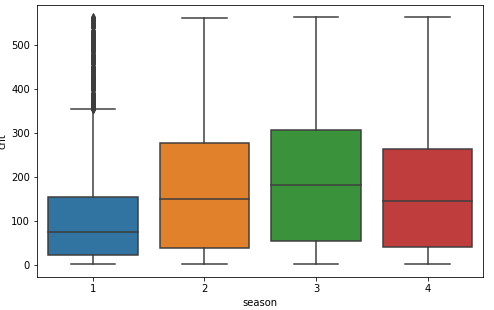
sns.boxplot("mnth", "cnt", data=inp2****

Looks like end of winter/ early spring months have the least bike riding instances.

1. **Make boxplot for cnt vs season**
   1. ***Which season has the highest rides in general? Expected?***

plt.figure(figsize=[10,6])

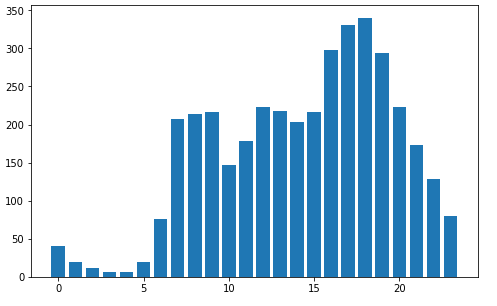
sns.boxplot("season", "cnt", data=inp2)

****

1. **Make a bar plot with the median value of cnt for each hr**
   1. ***Does this paint a different picture than the box plot?***

plt.figure(figsize=[8,5])

plt.bar(res.keys(), res.values)

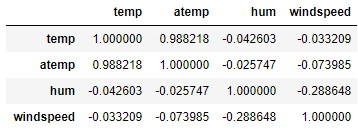
****

Paints a similar picture to the boxplot. Although the view is much cleaner and the pattern comes out much easier.

1. **Make a correlation matrix for variables – atemp, temp, hum, windspeed**
   1. ***Which variables have the highest correlation?***

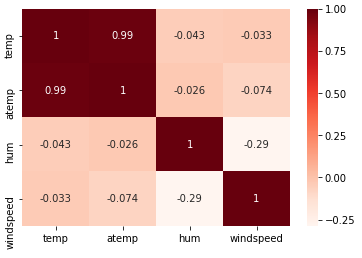
num\_vars = ['temp', 'atemp', 'hum', 'windspeed']

corrs = inp2[num\_vars].corr()

****

**Bonus: Heatmap of the correlations**

sns.heatmap(corrs, annot=True, cmap="Reds")

****

**8. Data pre-processing**

**A few key considerations for the pre-processing –**

**We seem to have plenty of categorical features. Since these categorical features can’t be used in the predictive model, we need to convert to a suitable numerical representation. Instead of creating dozens of new dummy variables, we will try to club levels of categorical features wherever possible. For a feature with high number of categorical levels, we can club the values that are very similar in value for the target variable**

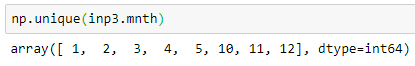
**First, create a copy of the dataframe into inp3**

1. **Treating ‘mnth’ column**
   1. **For values 5,6,7,8,9,10 – replace with a single value 5. This is because these have very similar values for cnt.**
   2. **Get dummies for the updated 6 ‘mnth’ values**

inp3 = inp2.copy()

inp3.mnth[inp3.mnth.isin([5,6,7,8,9])] = 5

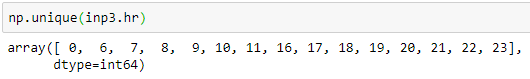
np.unique(inp3.mnth)



1. **Treating ‘hr’ column**
   1. **Create new mapping: 0-5: 0, 11-15: 11, other values are untouched. Again, the bucketing is done in a way that hr values with similar levels of cnt are treated the same.**

inp3.hr[inp3.hr.isin([0,1,2,3,4,5])] = 0

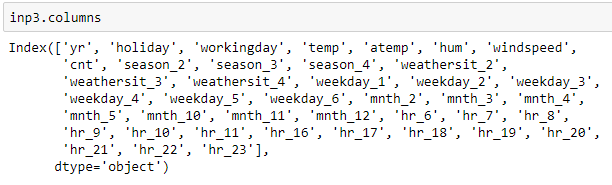
inp3.hr[inp3.hr.isin([11,12,13,14,15])] = 11

****

1. **Get dummy columns for season, weathersit, weekday, mnth, hr. We needn’t club these further, because as seen from the box plots, the levels seem to have different values for the median cnt.**

cat\_cols = ['season', 'weathersit', 'weekday', 'mnth', 'hr']

inp3 = pd.get\_dummies(inp3, columns=cat\_cols, drop\_first=True)



**9. Train test split – apply 70-30 split**

**- call the new dataframes df\_train, df\_test**

from sklearn.model\_selection import train\_test\_split

df\_train, df\_test = train\_test\_split(inp3, train\_size = 0.7, random\_state = 100)

**10. Separate X and Y for df\_train and df\_test. For example – you should have X\_train, y\_train from df\_train. y\_train should be the cnt column from inp3, X\_train should be all other columns.**

y\_train = df\_train.pop("cnt")

X\_train = df\_train

y\_test = df\_test.pop("cnt")

X\_test = df\_test

**10 . Model building**

* **Use Linear regression as the technique**
* **Report the R2 on the train set**

from sklearn.linear\_model import LinearRegression

lr = LinearRegression()

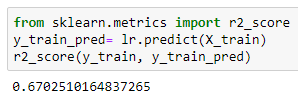
lr.fit(X\_train, y\_train)

Reporting r2 for the model

from sklearn.metrics import r2\_score

y\_train\_pred= lr.predict(X\_train)

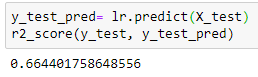
r2\_score(y\_train, y\_train\_pred)

****

**11. Make predictions on test set, report R2**

y\_test\_pred= lr.predict(X\_test)

r2\_score(y\_test, y\_test\_pred)

****