Time Series Capstone Project

January 19, 2019

1 Capstone Project Data Science

1.1 Project Title: Time series Analysis :JetRail Development and Investment Prediction

Problem Statement " Unicorn Investors wants to make an investment in a new form of transportation - JetRail. JetRail uses Jet propulsion technology and Foursquare Location Data (API) to run rails and move people at a high speed! The investment would only make sense, if they can get more than 1 Million monthly users with in next 18 months. In order to help Unicorn Ventures in their decision, you need to forecast the traffic on JetRail for the next 7 months. You are provided with traffic data of JetRail since inception in the test file."

Hypothesis Generation We will start with the first step, i.e. Hypothesis Generation. Hypothesis Generation is the process of listing out all the possible factors that can affect the outcome.

Hypothesis generation is done before having a look at the data in order to avoid any bias that may result after the observation.

1) Hypothesis Generation Hypothesis generation helps us to point out the factors which might affect our dependent variable. Below are some of the hypotheses which I think can affect the passenger count(dependent variable for this time series problem) on the JetRail:

There will be an increase in the traffic as the years pass by. Explanation - Population has a general upward trend with time, so I can expect more people to travel by JetRail. Also, generally companies expand their businesses over time leading to more customers travelling through JetRail.

The traffic will be high from May to October. Explanation - Tourist visits generally increases during this time perion.

Traffic on weekdays will be more as compared to weekends/holidays. Explanation - People will go to office on weekdays and hence the traffic will be more

Traffic during the peak hours will be high. Explanation - People will travel to work, college.

1.2 Getting the system ready and Loading Data

After building hypothesis we will try to validate our hypothesis result by taking first step

```
In [3]: import pandas as pd
        import numpy as np          # For mathematical calculations
        import matplotlib.pyplot as plt  # For plotting graphs
        from datetime import datetime    # To access datetime
        from pandas import Series        # To work on series
        %matplotlib inline
        import warnings                   # To ignore the warnings
        warnings.filterwarnings("ignore")

In [4]: train=pd.read_csv("Train_SU63ISt.csv")
        test=pd.read_csv("Test_0qrQsBZ.csv")

In [5]: train_original=train.copy()
        test_original=test.copy()
```

Data loaded and original copy is saved to another dateframe so as to preserve the datasets structure

1.3 Dataset Structure and Content

lets dive deep into data know its structure and what does dataset trying to convey

```
In [6]: train.columns,test.columns

Out[6]: (Index(['ID', 'Datetime', 'Count'], dtype='object'),
         Index(['ID', 'Datetime'], dtype='object'))
```

Three Features are availaable namely ID ,Datetime ,and count .

by the way one thing to ne noticed is that there is only two features in column set of test file . we have to determine traffick count

```
In [7]: train.duplicated().sum()

Out[7]: 0

In [8]: train.dtypes,test.dtypes

Out[8]: (ID           int64
         Datetime    object
         Count        int64
         dtype: object, ID           int64
         Datetime    object
         dtype: object)
```

here in train file two columns are nueric and one is object data i,e; string object we need convert to to datetime

```
In [9]: train.shape,test.shape

Out[9]: ((18288, 3), (5112, 2))
```

Dateset size has been obtained !!!

2 Feature Engineering

It involves Art of feature Engineering Feature selection feature Extraction

```
In [10]: train['Datetime'] = pd.to_datetime(train.Datetime,format='%d-%m-%Y %H:%M')
         test['Datetime'] = pd.to_datetime(test.Datetime,format='%d-%m-%Y %H:%M')
         test_original['Datetime'] = pd.to_datetime(test_original.Datetime,format='%d-%m-%Y %H
         train_original['Datetime'] = pd.to_datetime(train_original.Datetime,format='%d-%m-%Y
```

We made some hypothesis for the effect of hour, day, month and year on the passenger count. So, let's extract the year, month, day and hour from the Datetime to validate our hypothesis.

```
In [11]: for i in (train, test, test_original, train_original):
             i['year']=i.Datetime.dt.year
             i['month']=i.Datetime.dt.month
             i['day']=i.Datetime.dt.day
             i['Hour']=i.Datetime.dt.hour
```

We made a hypothesis for the traffic pattern on weekday and weekend as well. So, let's make a weekend variable to visualize the impact of weekend on traffic.

We will first extract the day of week from Datetime and then based on the values we will assign whether the day is a weekend or not.

Values of 5 and 6 represents that the days are weekend.

```
In [12]: train['day of week']=train['Datetime'].dt.dayofweek
         temp = train['Datetime']
         temp

Out[12]: 0        2012-08-25 00:00:00
         1        2012-08-25 01:00:00
         2        2012-08-25 02:00:00
         3        2012-08-25 03:00:00
         4        2012-08-25 04:00:00
         5        2012-08-25 05:00:00
         6        2012-08-25 06:00:00
         7        2012-08-25 07:00:00
         8        2012-08-25 08:00:00
         9        2012-08-25 09:00:00
         10       2012-08-25 10:00:00
         11       2012-08-25 11:00:00
         12       2012-08-25 12:00:00
         13       2012-08-25 13:00:00
         14       2012-08-25 14:00:00
         15       2012-08-25 15:00:00
         16       2012-08-25 16:00:00
         17       2012-08-25 17:00:00
         18       2012-08-25 18:00:00
         19       2012-08-25 19:00:00
         20       2012-08-25 20:00:00
```

```
21      2012-08-25 21:00:00
22      2012-08-25 22:00:00
23      2012-08-25 23:00:00
24      2012-08-26 00:00:00
25      2012-08-26 01:00:00
26      2012-08-26 02:00:00
27      2012-08-26 03:00:00
28      2012-08-26 04:00:00
29      2012-08-26 05:00:00
               ...
18258   2014-09-24 18:00:00
18259   2014-09-24 19:00:00
18260   2014-09-24 20:00:00
18261   2014-09-24 21:00:00
18262   2014-09-24 22:00:00
18263   2014-09-24 23:00:00
18264   2014-09-25 00:00:00
18265   2014-09-25 01:00:00
18266   2014-09-25 02:00:00
18267   2014-09-25 03:00:00
18268   2014-09-25 04:00:00
18269   2014-09-25 05:00:00
18270   2014-09-25 06:00:00
18271   2014-09-25 07:00:00
18272   2014-09-25 08:00:00
18273   2014-09-25 09:00:00
18274   2014-09-25 10:00:00
18275   2014-09-25 11:00:00
18276   2014-09-25 12:00:00
18277   2014-09-25 13:00:00
18278   2014-09-25 14:00:00
18279   2014-09-25 15:00:00
18280   2014-09-25 16:00:00
18281   2014-09-25 17:00:00
18282   2014-09-25 18:00:00
18283   2014-09-25 19:00:00
18284   2014-09-25 20:00:00
18285   2014-09-25 21:00:00
18286   2014-09-25 22:00:00
18287   2014-09-25 23:00:00
Name: Datetime, Length: 18288, dtype: datetime64[ns]
```

In [13]:
```python
def applyer(row):
    if row.dayofweek == 5 or row.dayofweek == 6:
        return 1
    else:
        return 0
```

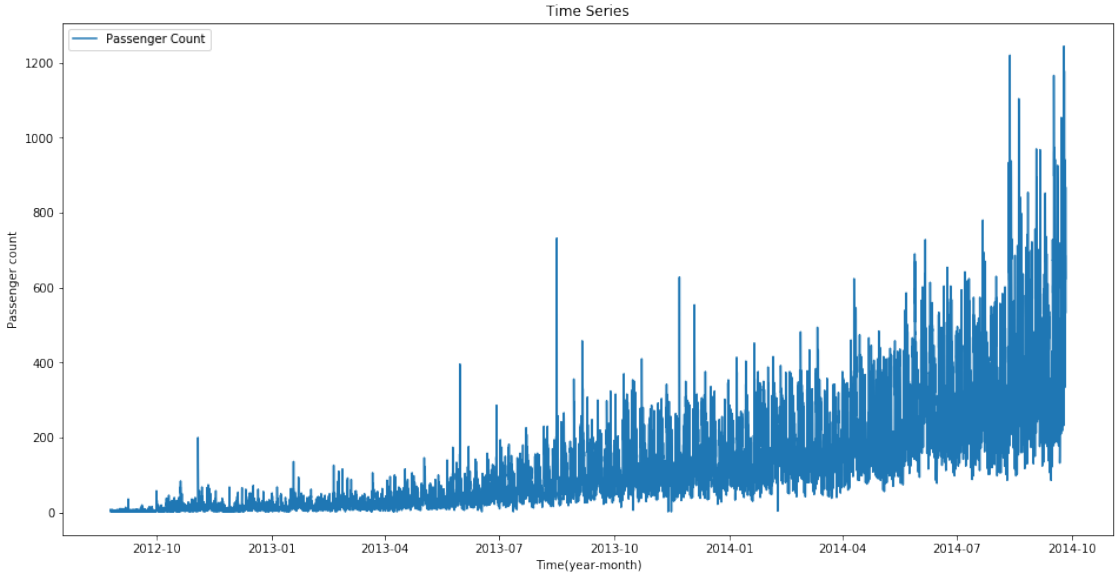
```
        temp2 = train['Datetime'].apply(applyer)
        train['weekend']=temp2
```

```
In [14]: print ('letss look at the time series')
```

```
letss look at the time series
```

```
In [15]: train.index = train['Datetime'] # indexing the Datetime to get the time period on the
         df=train.drop('ID',1)           # drop ID variable to get only the Datetime on x-axis
         ts = df['Count']
         plt.figure(figsize=(16,8))
         plt.plot(ts, label='Passenger Count')
         plt.title('Time Series')
         plt.xlabel("Time(year-month)")
         plt.ylabel("Passenger count")
         plt.legend(loc='best')
```

```
Out[15]: <matplotlib.legend.Legend at 0x24b5c913f28>
```



Here we can infer that there is an increasing trend in the series, i.e., the number of count is increasing with respect to time. We can also see that at certain points there is a sudden increase in the number of counts. The possible reason behind this could be that on particular day, due to some event the traffic was high.

We will work on the train file for all the analysis and will use the test file for forecasting.

2.1 Explorotary Data Analysis

Let's Reacall our hyporthesis made ,whih can affect our outcome or rsult 1.year passess by 2.month of the year 3.day of the month 4.Hour in a day which has peak load 5.day of week: either weekdays or weekends

In [16]: train

Out[16]:

Datetime	ID	Datetime	Count	year	month	day	Hour
2012-08-25 00:00:00	0	2012-08-25 00:00:00	8	2012	8	25	0
2012-08-25 01:00:00	1	2012-08-25 01:00:00	2	2012	8	25	1
2012-08-25 02:00:00	2	2012-08-25 02:00:00	6	2012	8	25	2
2012-08-25 03:00:00	3	2012-08-25 03:00:00	2	2012	8	25	3
2012-08-25 04:00:00	4	2012-08-25 04:00:00	2	2012	8	25	4
2012-08-25 05:00:00	5	2012-08-25 05:00:00	2	2012	8	25	5
2012-08-25 06:00:00	6	2012-08-25 06:00:00	2	2012	8	25	6
2012-08-25 07:00:00	7	2012-08-25 07:00:00	2	2012	8	25	7
2012-08-25 08:00:00	8	2012-08-25 08:00:00	6	2012	8	25	8
2012-08-25 09:00:00	9	2012-08-25 09:00:00	2	2012	8	25	9
2012-08-25 10:00:00	10	2012-08-25 10:00:00	2	2012	8	25	10
2012-08-25 11:00:00	11	2012-08-25 11:00:00	6	2012	8	25	11
2012-08-25 12:00:00	12	2012-08-25 12:00:00	4	2012	8	25	12
2012-08-25 13:00:00	13	2012-08-25 13:00:00	2	2012	8	25	13
2012-08-25 14:00:00	14	2012-08-25 14:00:00	6	2012	8	25	14
2012-08-25 15:00:00	15	2012-08-25 15:00:00	2	2012	8	25	15
2012-08-25 16:00:00	16	2012-08-25 16:00:00	2	2012	8	25	16
2012-08-25 17:00:00	17	2012-08-25 17:00:00	2	2012	8	25	17
2012-08-25 18:00:00	18	2012-08-25 18:00:00	2	2012	8	25	18
2012-08-25 19:00:00	19	2012-08-25 19:00:00	2	2012	8	25	19
2012-08-25 20:00:00	20	2012-08-25 20:00:00	2	2012	8	25	20
2012-08-25 21:00:00	21	2012-08-25 21:00:00	6	2012	8	25	21
2012-08-25 22:00:00	22	2012-08-25 22:00:00	2	2012	8	25	22
2012-08-25 23:00:00	23	2012-08-25 23:00:00	2	2012	8	25	23
2012-08-26 00:00:00	24	2012-08-26 00:00:00	4	2012	8	26	0
2012-08-26 01:00:00	25	2012-08-26 01:00:00	6	2012	8	26	1
2012-08-26 02:00:00	26	2012-08-26 02:00:00	2	2012	8	26	2
2012-08-26 03:00:00	27	2012-08-26 03:00:00	4	2012	8	26	3
2012-08-26 04:00:00	28	2012-08-26 04:00:00	2	2012	8	26	4
2012-08-26 05:00:00	29	2012-08-26 05:00:00	2	2012	8	26	5
...	...	...	...	...	...	...	...
2014-09-24 18:00:00	18258	2014-09-24 18:00:00	966	2014	9	24	18
2014-09-24 19:00:00	18259	2014-09-24 19:00:00	1178	2014	9	24	19
2014-09-24 20:00:00	18260	2014-09-24 20:00:00	1008	2014	9	24	20
2014-09-24 21:00:00	18261	2014-09-24 21:00:00	936	2014	9	24	21
2014-09-24 22:00:00	18262	2014-09-24 22:00:00	848	2014	9	24	22
2014-09-24 23:00:00	18263	2014-09-24 23:00:00	942	2014	9	24	23
2014-09-25 00:00:00	18264	2014-09-25 00:00:00	686	2014	9	25	0
2014-09-25 01:00:00	18265	2014-09-25 01:00:00	546	2014	9	25	1
2014-09-25 02:00:00	18266	2014-09-25 02:00:00	550	2014	9	25	2
2014-09-25 03:00:00	18267	2014-09-25 03:00:00	416	2014	9	25	3
2014-09-25 04:00:00	18268	2014-09-25 04:00:00	344	2014	9	25	4
2014-09-25 05:00:00	18269	2014-09-25 05:00:00	334	2014	9	25	5
2014-09-25 06:00:00	18270	2014-09-25 06:00:00	416	2014	9	25	6

2014-09-25 07:00:00	18271	2014-09-25 07:00:00	498	2014	9	25	7
2014-09-25 08:00:00	18272	2014-09-25 08:00:00	482	2014	9	25	8
2014-09-25 09:00:00	18273	2014-09-25 09:00:00	512	2014	9	25	9
2014-09-25 10:00:00	18274	2014-09-25 10:00:00	724	2014	9	25	10
2014-09-25 11:00:00	18275	2014-09-25 11:00:00	656	2014	9	25	11
2014-09-25 12:00:00	18276	2014-09-25 12:00:00	672	2014	9	25	12
2014-09-25 13:00:00	18277	2014-09-25 13:00:00	642	2014	9	25	13
2014-09-25 14:00:00	18278	2014-09-25 14:00:00	616	2014	9	25	14
2014-09-25 15:00:00	18279	2014-09-25 15:00:00	686	2014	9	25	15
2014-09-25 16:00:00	18280	2014-09-25 16:00:00	654	2014	9	25	16
2014-09-25 17:00:00	18281	2014-09-25 17:00:00	622	2014	9	25	17
2014-09-25 18:00:00	18282	2014-09-25 18:00:00	680	2014	9	25	18
2014-09-25 19:00:00	18283	2014-09-25 19:00:00	868	2014	9	25	19
2014-09-25 20:00:00	18284	2014-09-25 20:00:00	732	2014	9	25	20
2014-09-25 21:00:00	18285	2014-09-25 21:00:00	702	2014	9	25	21
2014-09-25 22:00:00	18286	2014-09-25 22:00:00	580	2014	9	25	22
2014-09-25 23:00:00	18287	2014-09-25 23:00:00	534	2014	9	25	23

Datetime	day of week	weekend
2012-08-25 00:00:00	5	1
2012-08-25 01:00:00	5	1
2012-08-25 02:00:00	5	1
2012-08-25 03:00:00	5	1
2012-08-25 04:00:00	5	1
2012-08-25 05:00:00	5	1
2012-08-25 06:00:00	5	1
2012-08-25 07:00:00	5	1
2012-08-25 08:00:00	5	1
2012-08-25 09:00:00	5	1
2012-08-25 10:00:00	5	1
2012-08-25 11:00:00	5	1
2012-08-25 12:00:00	5	1
2012-08-25 13:00:00	5	1
2012-08-25 14:00:00	5	1
2012-08-25 15:00:00	5	1
2012-08-25 16:00:00	5	1
2012-08-25 17:00:00	5	1
2012-08-25 18:00:00	5	1
2012-08-25 19:00:00	5	1
2012-08-25 20:00:00	5	1
2012-08-25 21:00:00	5	1
2012-08-25 22:00:00	5	1
2012-08-25 23:00:00	5	1
2012-08-26 00:00:00	6	1
2012-08-26 01:00:00	6	1
2012-08-26 02:00:00	6	1
2012-08-26 03:00:00	6	1

```
2012-08-26 04:00:00            6        1
2012-08-26 05:00:00            6        1
...                          ...      ...
2014-09-24 18:00:00            2        0
2014-09-24 19:00:00            2        0
2014-09-24 20:00:00            2        0
2014-09-24 21:00:00            2        0
2014-09-24 22:00:00            2        0
2014-09-24 23:00:00            2        0
2014-09-25 00:00:00            3        0
2014-09-25 01:00:00            3        0
2014-09-25 02:00:00            3        0
2014-09-25 03:00:00            3        0
2014-09-25 04:00:00            3        0
2014-09-25 05:00:00            3        0
2014-09-25 06:00:00            3        0
2014-09-25 07:00:00            3        0
2014-09-25 08:00:00            3        0
2014-09-25 09:00:00            3        0
2014-09-25 10:00:00            3        0
2014-09-25 11:00:00            3        0
2014-09-25 12:00:00            3        0
2014-09-25 13:00:00            3        0
2014-09-25 14:00:00            3        0
2014-09-25 15:00:00            3        0
2014-09-25 16:00:00            3        0
2014-09-25 17:00:00            3        0
2014-09-25 18:00:00            3        0
2014-09-25 19:00:00            3        0
2014-09-25 20:00:00            3        0
2014-09-25 21:00:00            3        0
2014-09-25 22:00:00            3        0
2014-09-25 23:00:00            3        0

[18288 rows x 9 columns]
```

```
In [17]: train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 18288 entries, 2012-08-25 00:00:00 to 2014-09-25 23:00:00
Data columns (total 9 columns):
ID             18288 non-null int64
Datetime       18288 non-null datetime64[ns]
Count          18288 non-null int64
year           18288 non-null int64
month          18288 non-null int64
day            18288 non-null int64
Hour           18288 non-null int64
```

```
day of week    18288 non-null int64
weekend        18288 non-null int64
dtypes: datetime64[ns](1), int64(8)
memory usage: 1.4 MB
```

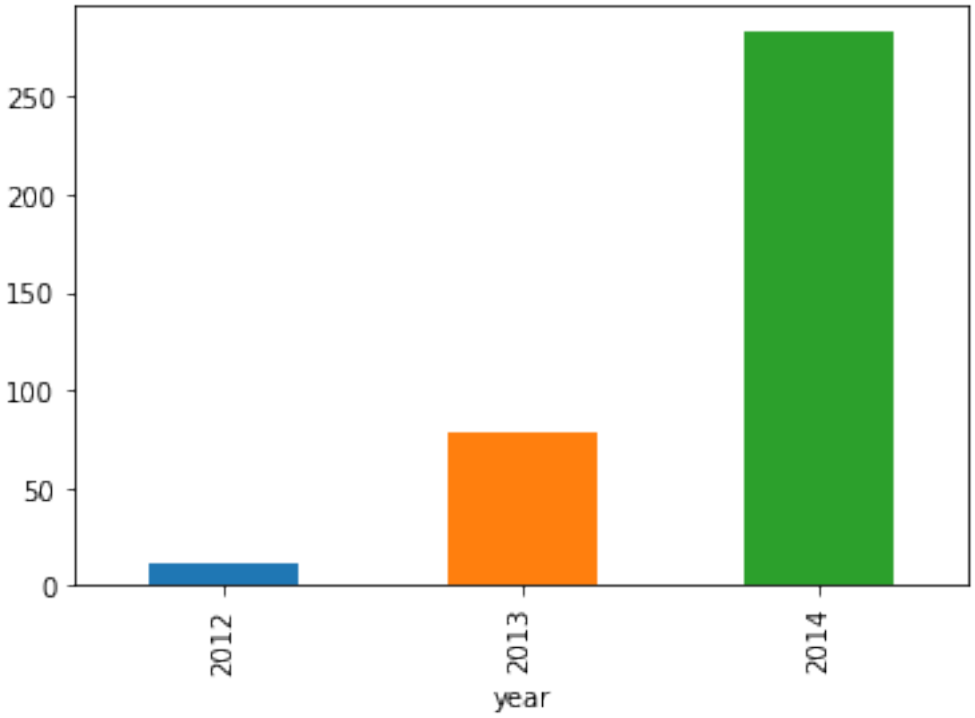
```
In [18]: train.describe(include='all')
```

Out[18]:

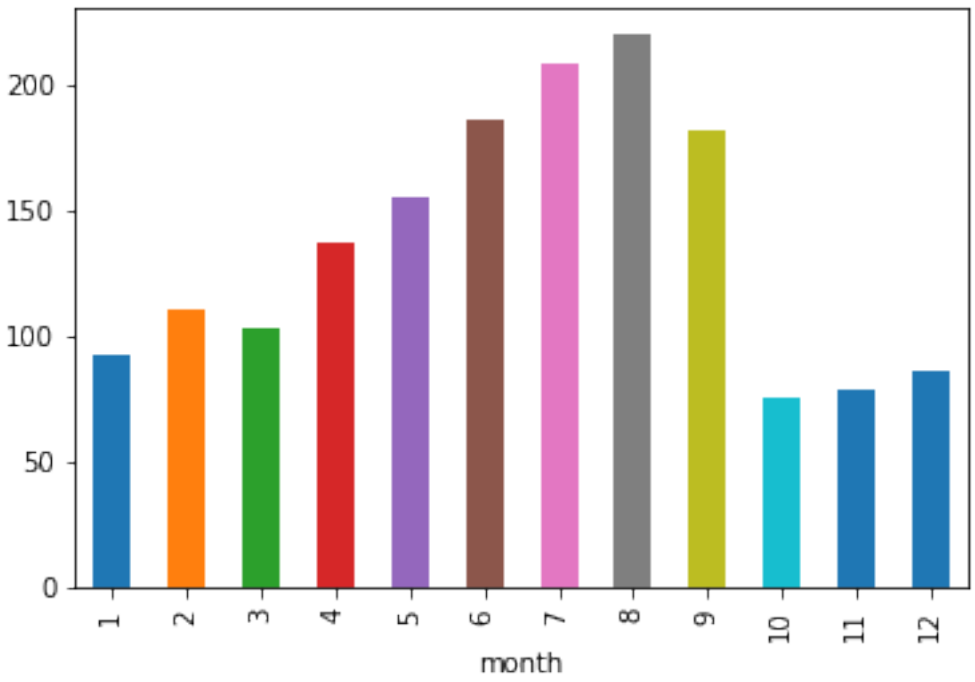
	ID	Datetime	Count	year	month	day	Hour	day of week	weekend
count	18288.000000	18288	18288.000000	18288.000000	18288.000000	18288.000000	18288.000000	18288.000000	18288.000000
unique	NaN	18288	NaN	NaN	NaN	NaN	NaN	NaN	NaN
top	NaN	2012-11-11 10:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN
freq	NaN	1	NaN	NaN	NaN	NaN	NaN	NaN	NaN
first	NaN	2012-08-25 00:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN
last	NaN	2014-09-25 23:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN
mean	9143.500000	NaN	138.958115	2013.182415	6.620735	15.744094	11.500000	2.998688	0.286089
std	5279.435197	NaN	153.467461	0.698390	3.406007	8.803358	6.922376	2.001038	0.451944
min	0.000000	NaN	2.000000	2012.000000	1.000000	1.000000	0.000000	0.000000	0.000000
25%	4571.750000	NaN	22.000000	2013.000000	4.000000	8.000000	5.750000	1.000000	0.000000
50%	9143.500000	NaN	84.000000	2013.000000	7.000000	16.000000	11.500000	3.000000	0.000000
75%	13715.250000	NaN	210.000000	2014.000000	9.000000	23.000000	17.250000	5.000000	1.000000
max	18287.000000	NaN	1244.000000	2014.000000	12.000000	31.000000	23.000000	6.000000	1.000000

```
In [19]: train.groupby('year')['Count'].mean().plot.bar()
```

```
Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0x24b5cad09e8>
```

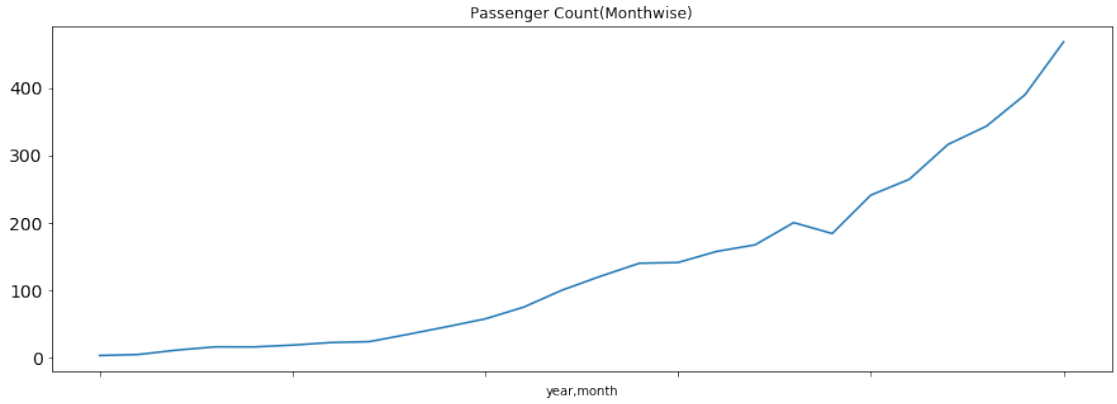



```
In [20]: train.groupby('month')['Count'].mean()

Out[20]: month
         1      92.817204
         2     111.196429
         3     103.791667
         4     137.680556
         5     155.055108
         6     186.723611
         7     209.071237
         8     220.497585
         9     181.958824
         10     75.459677
         11     78.413889
         12     86.533602
         Name: Count, dtype: float64

In [21]: train.groupby('month')['Count'].mean().plot.bar()

Out[21]: <matplotlib.axes._subplots.AxesSubplot at 0x24b5ca5c940>
```




```
In [22]: temp=train.groupby(['year', 'month'])['Count'].mean()
         temp.plot(figsize=(15,5), title= 'Passenger Count(Monthwise)', fontsize=14)
```

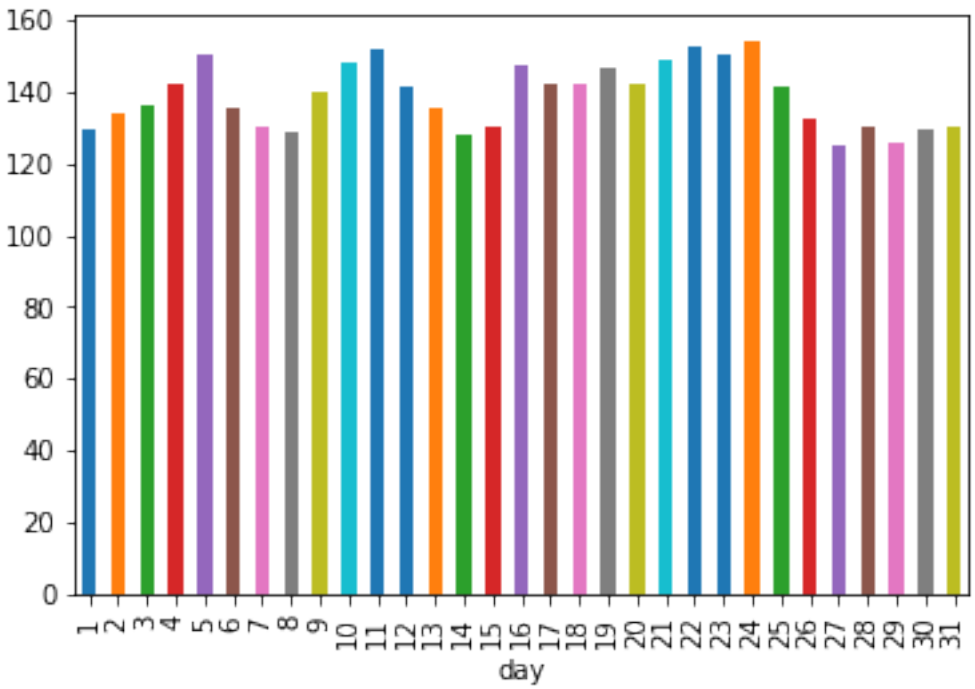
```
Out[22]: <matplotlib.axes._subplots.AxesSubplot at 0x24b5caa34e0>
```



here we ccan see that increasing trend if we combine the year and month !!

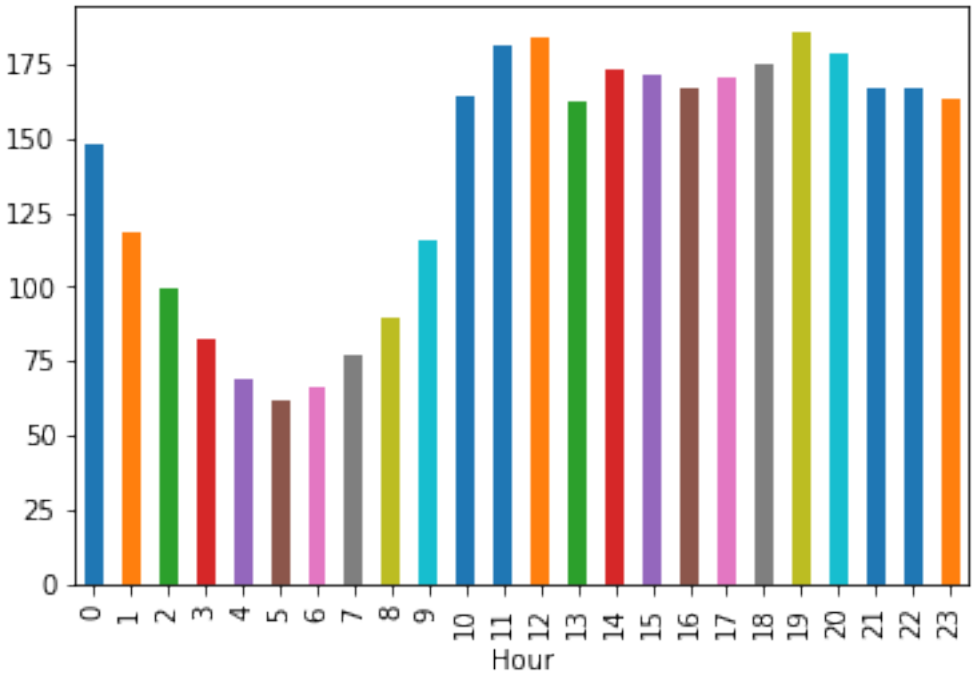
```
In [23]: train.groupby('day')['Count'].mean().plot.bar()
```

```
Out[23]: <matplotlib.axes._subplots.AxesSubplot at 0x24b5cf78c18>
```



In [24]: `train.groupby('Hour')['Count'].mean().plot.bar()`

Out[24]: `<matplotlib.axes._subplots.AxesSubplot at 0x24b5d07c588>`

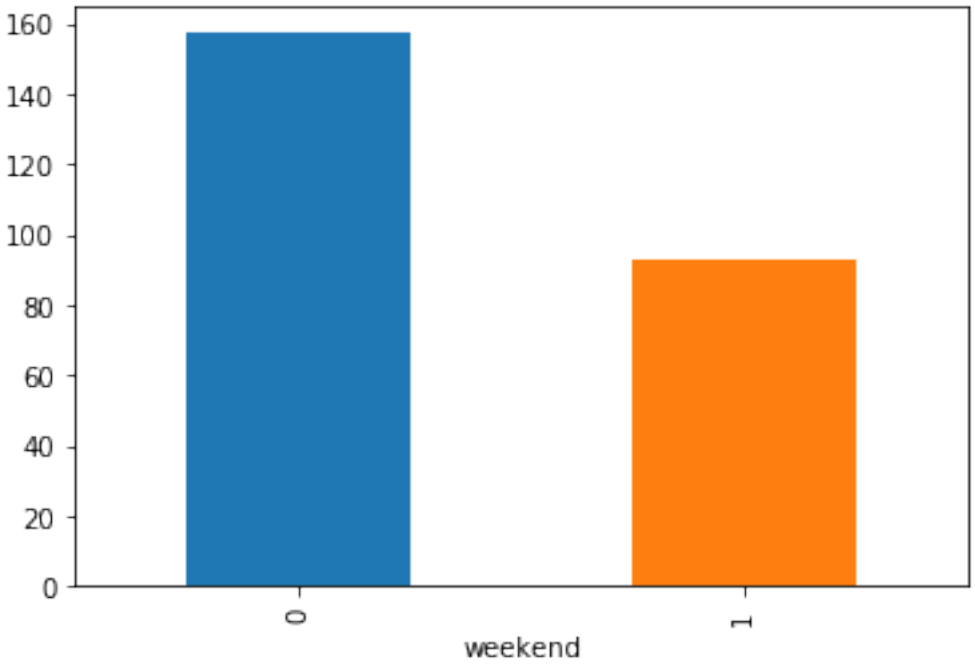


We are not getting much insights from day wise count of the passengers.

We also made a hypothesis that the traffic will be more during peak hours. So let's see the mean of hourly passenger count.

It can be seen that there is hourly incresinng trend from morning to evening and then to night

```
In [25]: train.groupby('weekend')['Count'].mean().plot.bar()

Out[25]: <matplotlib.axes._subplots.AxesSubplot at 0x24b5d100c18>
```



It can be inferred fromabobe barplot that on weekdays traffi count is more than weekends that is saturday and sunday.

now lets examine the trend on the basis of day of week from 0 to 6 here 0 is monday and 6 is sunday !

```
In [26]: train.groupby('day of week')['Count'].mean().plot.bar()

Out[26]: <matplotlib.axes._subplots.AxesSubplot at 0x24b5e2a1240>
```