Project — Bike rental count Prediction

Ayush Saini
Jun 23 · 18 min read

Chapter 1 Introduction

1.1 Problem Statement

The objective of this Case is to predict the number of bikes rent count based on environmental and seasonal settings.

We are provided with the daily data of bike renting count for two years (2010–2011) and we have to predict the number of bikes rented for a given seasonal and environmental settings.

1.2 Data Overview

	instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
0	1	2011-01-01	1	0	1	0	6	0	2	0.344167	0.363625	0.805833	0.160446	331	654	985
1	2	2011-01-02	1	0	1	0	0	0	2	0.363478	0.353739	0.696087	0.248539	131	670	801
2	3	2011-01-03	1	0	1	0	1	1	1	0.196364	0.189405	0.437273	0.248309	120	1229	1349
3	4	2011-01-04	1	0	1	0	2	1	1	0.200000	0.212122	0.590435	0.160296	108	1454	1562
4	5	2011-01-05	1	0	1	0	3	1	1	0.226957	0.229270	0.436957	0.186900	82	1518	1600

Fig1.1 First 5 rows of data

As we can see, there are 7 categorical variables and 7 numeric variables. The dependent variable or the target variable is cnt, i.e. the count of people renting bikes. We have a total of 16 columns, in which we have 13 independent variables and 3 dependent variables. ‘casual’, ‘registered’ and ‘cnt’ (dependent variables)are the counting of renting bikes for a particular day. Casual counting is for the non-registered customers, registered counting is for registered customers and cnt is for total counting i.e. casual + registered.

1.2.1 Detailed data attributes

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

yr: Year (0: 2011, 1:2012)

mnth: Month (1 to 12) (Jan to Dec)

holiday: whether the day is a holiday or not (extracted from Holiday Schedule)

weekday: Day of the week

workingday: If the day is neither weekend nor holiday is 1, otherwise is 0.

weathersit: (extracted from Freemeteo)
1: Clear, Few clouds, Partly cloudy, Partly cloudy
2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
3: Light Snow/Heavy snow, Heavy Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds

temp: Normalized temperature in Celsius. The values are derived via (t-t_min)/(t_max-t_min), t_min=-8, t_max=+39

atemp: Normalized feeling temperature in Celsius. The values are derived via (t-t_min)/(t_max- t_min), t_min=-16, t_max=+50

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

As we can see here we have a time-series dataset of two years (2010–2011), where we have different variables including environmental factors (Temperature, season, humidity, wind speed, etc) which can determine if a person will rent the bike on that day or not. We'll analyze each variable dependency on the target variable as well as the other variables and then we'll make our machine learning model on those variable to predict the bike rental count of a particular day.

Chapter 2 Data Interpretation and Visualisations

2.1 Exploratory Data Analysis

Exploratory Data Analysis refers to the critical process of performing initial investigations on data so as to discover patterns, to spot anomalies, to test hypothesis and to check assumptions with the help of summary statistics and graphical representations.

First, we take a look at the datatypes of the variables in the data

```
instant         int64
dteday         object
season          int64
yr              int64
mnth            int64
holiday         int64
weekday         int64
workingday      int64
weathersit      int64
temp          float64
atemp         float64
hum           float64
windspeed     float64
casual          int64
registered      int64
cnt             int64
dtype: object
```

Fig 2.1 Datatypes of the variables of the data

Now, Let's take a look number of unique values of each variable.

```
instant       731
dteday        731
season          4
yr              2
mnth           12
holiday         2
weekday         7
workingday      2
weathersit      3
temp          499
atemp         690
hum           595
windspeed     650
casual        606
registered    679
cnt           696
dtype: int64
```

Fig 2.2 Number of unique values of variables of the data

We have datatype as object for dteday and rest others have int and float.

Time-based variables: 'dteday', 'yr', 'mnth', 'season', 'weekday'
Now, what should be considered for them either continuous or categorical?

'**dteday**' has a date of every day, so all unique it could not be categorical.

'**yr**' we have two value 0 for 2010 and 1 for 2011, so we can consider it as categorical.

'**mnth**': can we consider a month like jan, 2010 and jan 2011 as single category? For a category condition need to be the same, like in a school two student are from 5th standard and their class category would be 5, i.e. same for both. We have bike renting dataset and situation for jan 2010 and jan 2011 would be different. So, with one point of view it could be same category for a month but with other points of the condition could not be same for jan 2010 and jan 2011. It has 12 values, so consider it as a category, will introduce 11 dimensions to our dataset. So, we will make bins for this column in the feature engineering section.

'**season**' has four unique values, so we would consider it as a category.

'**weekday**': Its same like mnth, so we will make bins for this in the feature engineering section.

Categorical: Holiday and working day having value 0 for non-holiday/non-working day 1 for the opposite. So it would be our categorical variable. 'weathersit': it containing 4 unique values, defining different four different conditions of weather. Conditions are based on Freemeteo. Our dataset containing only three conditions.

Continuous Variables*:* 'temp', 'atemp', 'hum', 'windspeed' are continuous values, and in our dataset, they are in a normalized format.

Target variables: 'casual', 'registered' and 'cnt' are our target variables and in continuous form. So our problem would be a regression problem.

Numerical Data Analysis

	temp	atemp	hum	windspeed	casual	registered	cnt
count	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000
mean	0.495385	0.474354	0.627894	0.190486	848.176471	3656.172367	4504.348837
std	0.183051	0.162961	0.142429	0.077498	686.622488	1560.256377	1937.211452
min	0.059130	0.079070	0.000000	0.022392	2.000000	20.000000	22.000000
25%	0.337083	0.337842	0.520000	0.134950	315.500000	2497.000000	3152.000000
50%	0.498333	0.486733	0.626667	0.180975	713.000000	3662.000000	4548.000000
75%	0.655417	0.608602	0.730209	0.233214	1096.000000	4776.500000	5956.000000
max	0.861667	0.840896	0.972500	0.507463	3410.000000	6946.000000	8714.000000

Fig 2.3 Summary of the numerical variables.

We have four independent continuous variables(temp, atemp, hum, windspeed) and three dependent continuous variables (casual, registered, cnt). As we can observe from the above table, all independent variables between 0 and 1. The dataset contains normalized values for all four columns. 'cnt' i.e. target variable has a minimum 22 counting and maximum 8714 countings.

	instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
0	1	2011-01-01	1	0	1	0	6	0	2	0.344167	0.363625	0.805833	0.160446	331	654	985
1	2	2011-01-02	1	0	1	0	0	0	2	0.363478	0.353739	0.696087	0.248539	131	670	801
2	3	2011-01-03	1	0	1	0	1	1	1	0.196364	0.189405	0.437273	0.248309	120	1229	1349
3	4	2011-01-04	1	0	1	0	2	1	1	0.200000	0.212122	0.590435	0.160296	108	1454	1562
4	5	2011-01-05	1	0	1	0	3	1	1	0.226957	0.229270	0.436957	0.186900	82	1518	1600

Fig 2.4 First 5 rows of data

Here, we also see that the sum of casual and registered bike rentals gives the 'cnt' (total count of the bikes rented).

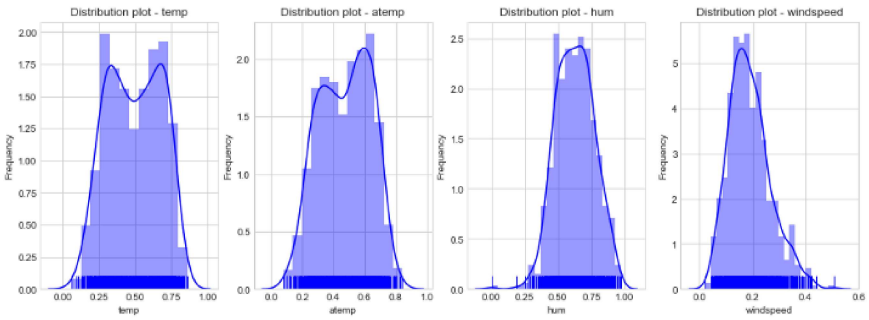


Fig2.5 Distribution of the independent numerical variables

Distribution is almost normal with little skewness for hum and windspeed and near to normal for temp and atemp with little bimodal effect.

Categorical Data Analysis

Now we will do the analysis on the categorical variables of the data.

```
season
3    188
2    184
1    181
4    178
Name: season, dtype: int64

yr
1    366
0    365
Name: yr, dtype: int64
```

```
mnth
12    62
10    62
8     62
7     62
5     62
3     62
1     62
11    60
9     60
6     60
4     60
2     57
Name: mnth, dtype: int64

holiday
0    710
1     21
Name: holiday, dtype: int64

weekday
6    105
1    105
0    105
5    104
4    104
3    104
2    104
Name: weekday, dtype: int64

workingday
1    500
0    231
Name: workingday, dtype: int64

weathersit
1    463
2    247
3     21
Name: weathersit, dtype: int64
```

Here we observe that the holiday variable is highly imbalanced with 21 holidays and 710 non-holidays which may prove to be non-important for our model development. We'll be looking forward to this in feature selection and engineering section.
Whereas, weathersit has no values for 4th category (Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog).

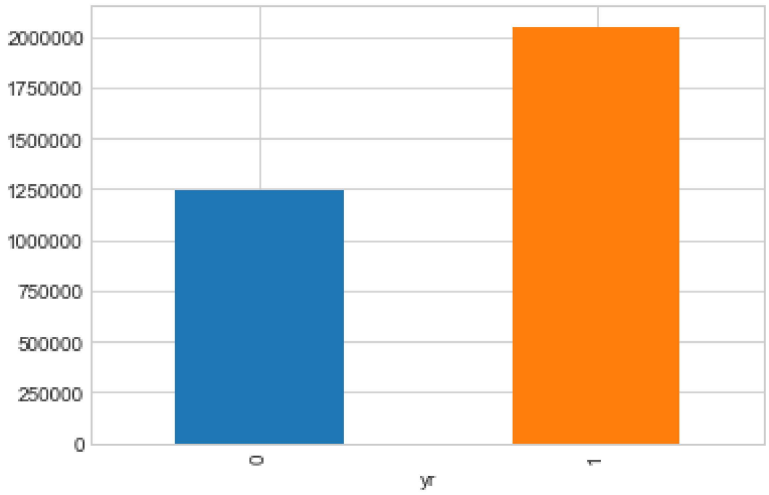


Fig 2.6 Number of total bike rentals each year

Here we see there are more Bike rentals in 2011 as compared to 2010. Now we'll see if all the other categorical variables follow the same pattern for both years.

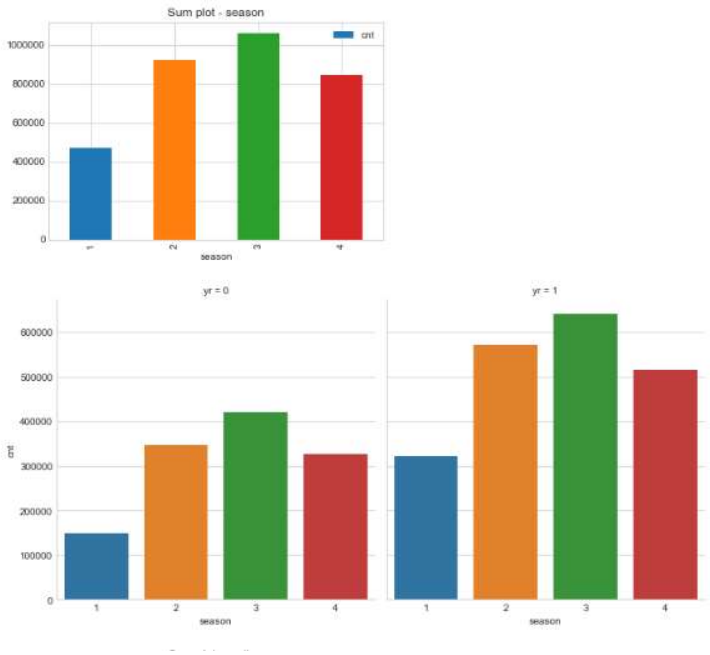


Fig 2.8 season-wise plot vs total bike rental count

These plots show that people prefer to rent bikes mostly in fall season and least in the spring season and follow the same pattern for both years.

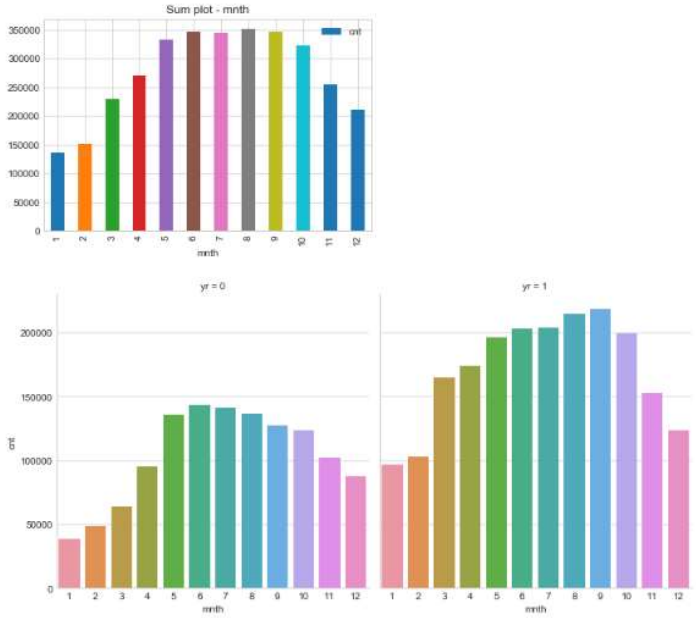


Fig 2.9 month-wise plot vs total count of bike rentals

The Visualisations shows that an increase in the number of bike rentals in June, July, and August and following a decrease in the months of Winter season.

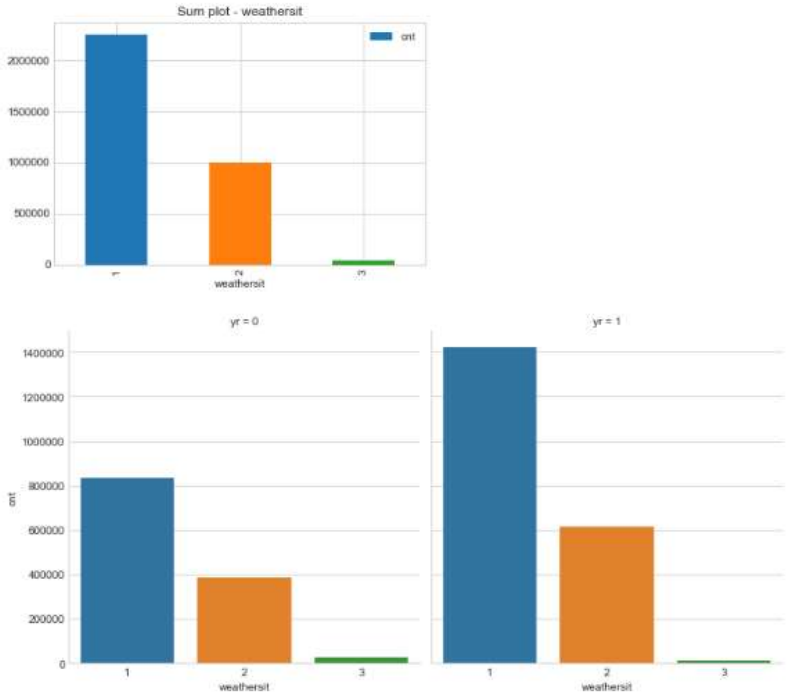


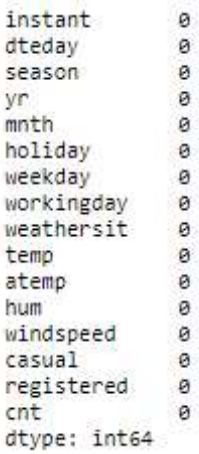
Fig 2.10 weather-wise plot vs total count of bike rentals

These visualizations clearly show that people prefer to rent a bike when the weather is clear and least when the weather is snowy and it is raining heavily.

Conclusion

So, the conclusion from the visualizations and data analysis is that people prefer to rent a bike when the sky is clear or a little cloudy and most in the fall season. Weekdays/Weekends doesn't really affect the bike rental count.

Chapter 3 Missing Value Analysis



```
instant       0
dteday        0
season        0
yr            0
mnth          0
holiday       0
weekday       0
workingday    0
weathersit    0
temp          0
atemp         0
hum           0
windspeed     0
casual        0
registered    0
cnt           0
dtype: int64
```

Fig 3.1 Output showing there are no outliers in the data

Missing value analysis helps address several concerns caused by incomplete data. If cases with missing values are systematically different from cases without missing values, the results can be misleading.
These values can be computed by various methods like KNN, mean, median, linear regression, etc.

But here in our case luckily, there are no missing values, Therefore, we don't have to impute missing values.

Chapter 4 Outlier Analysis

Outlier detection and treatment is always a tricky part especially when our dataset is small. The box plot method detects outlier if any value is greater than (Q3 + (1.5 * IQR)) or less than (Q1—(1.5 * IQR)).

```
Q1 > 25% of data are less than or equal to this value
Q2 or Median -> 50% of data are less than or equal to this
value
Q3 > 75% of data are less than or equal to this value
IQR(Inter Quartile Range) = Q3 — Q1
```

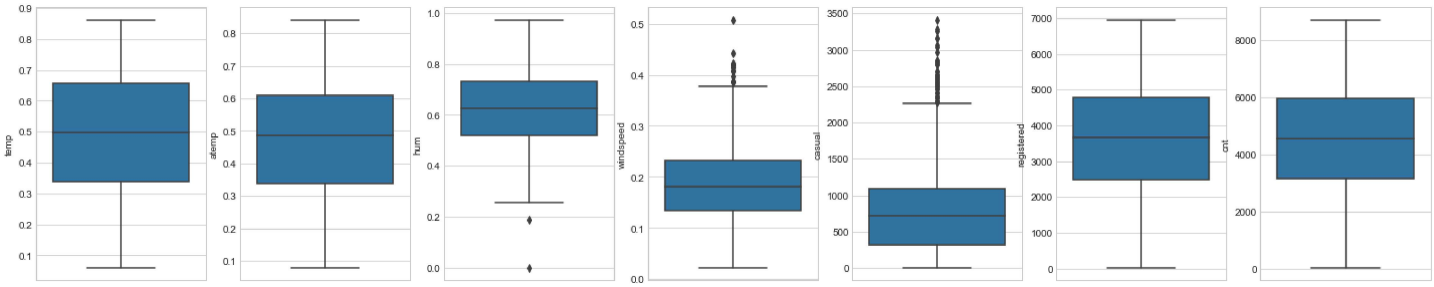


Fig 4.1 Boxplot to visualize outliers

The visualizations show no outliers in temp and atemp variables with few in humidity and quite a few in windspeed variable.

Usually, in the boxplot method, it only removes the 1% data from the whole dataset which seems to be an anomaly in the data. We assume that since the data is too small, it won't affect our whole dataset and modeling process.

But in this case, our dataset is based on season and environmental condition. Suppose for clear weather condition, windspeed is normal maximum time. Now someday windspeed is higher than other day and it has a high value. Now, Boxplot may consider it as an outlier, as the distance from median would be high for this value and due to high windspeed, there may be a decrease in bike rental counting for that day. So, the boxplot method would remove that data point, but that data point could be an important predictor.

So, removing outliers in these case won't be a good idea as we might be removing some important data from our dataset.

Chapter 5 Feature Selection and Feature Engineering

5.1 Correlation Analysis

Firstly, we will plot the multicollinearity for our continuous variables using scatter-plots.

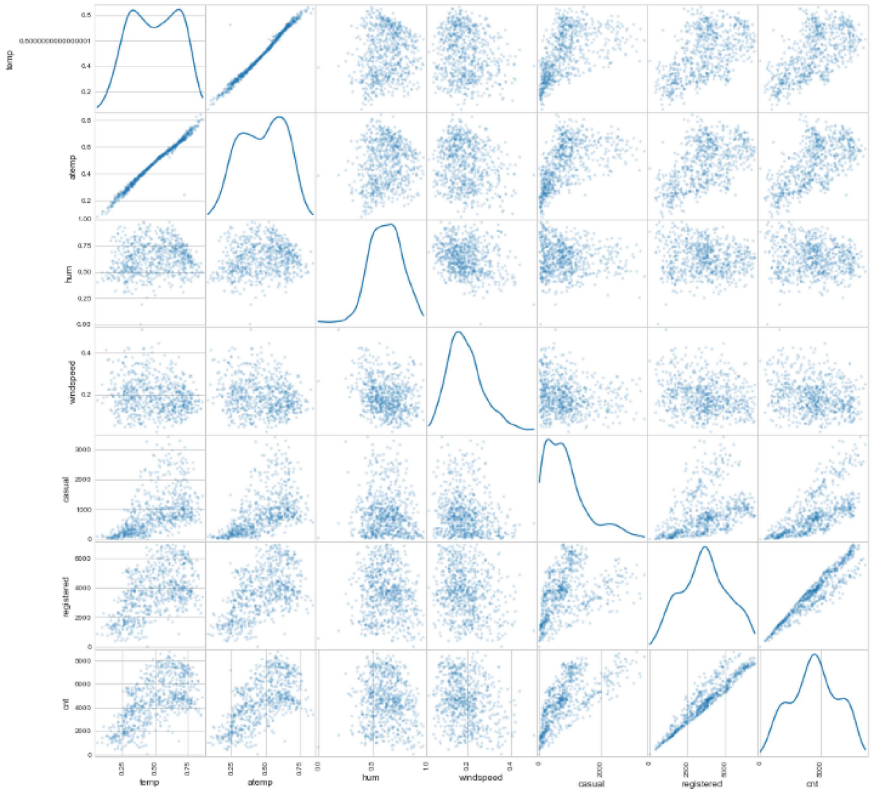


Fig 5.1 Scatter plot of numerical continuous variables.

These plots show collinearity between
> temp and atemp,
> cnt and registered and casual

Now again we plot the heatmap of the correlation between these variables.

	temp	atemp	hum	windspeed	casual	registered	cnt
temp	1	0.991702	0.126963	-0.157944	0.543285	0.540012	0.627494
atemp	0.991702	1	0.139988	-0.183643	0.543864	0.544192	0.631066
hum	0.126963	0.139988	1	-0.248489	-0.0770079	-0.0910886	-0.100659
windspeed	-0.157944	-0.183643	-0.248489	1	-0.167613	-0.217449	-0.234545
casual	0.543285	0.543864	-0.0770079	-0.167613	1	0.395282	0.672804
registered	0.540012	0.544192	-0.0910886	-0.217449	0.395282	1	0.945517
cnt	0.627494	0.631066	-0.100659	-0.234545	0.672804	0.945517	1

Fig 5.2 heatmap of correlation between the continuous variables

From heatmap and correlation plot we can see that there is multicollinearity present between temp and atemp, so we will drop one of those columns as per the importance of features. Multicollinearity present between registered and cnt, but these are our target variable. We will only use total counting i.e. cnt as our target variable and would drop casual and registered column.

5.2 Chi-square test

For getting dependency between categorical variables we would use chi-square test of independence test. Basically, we have mnth , yr, weekday, holiday, workingday, weathersit and season as categorical variables.

So, we want our categorical variable should be independent. If we get p-value less than 0.05, it means we need to reject the null hypothesis and accept the alternate hypothesis which means variables are dependent. So, we would check for every combination and would print p-value.

Null Hypothesis for Chi-square test of Independence: Two variables are independent.

Alternate Hypothesis: Two variables are not independent.

	season	yr	mnth	holiday	weekday	workingday	weathersit
season	-	1.0	0.0	0.683	1.0	0.887	0.021
yr	1.0	-	1.0	0.995	1.0	0.98	0.127
mnth	0.0	1.0	-	0.559	1.0	0.993	0.015
holiday	0.683	0.995	0.559	-	0.0	0.0	0.601
weekday	1.0	1.0	1.0	0.0	-	0.0	0.278
workingday	0.887	0.98	0.993	0.0	0.0	-	0.254
weathersit	0.021	0.127	0.015	0.601	0.278	0.254	-

Fig 5.3 Chi-square test

Here, from the table, we can see some values are less than 0.05 indicating them they are dependent. Holiday is showing collinearity with week and workingday. Month is showing collinearity with season. Weathersit showing collinearity with the season.

5.3 ANOVA Analysis

ANOVA is performed with numeric and categorical data combined. The PR(>F) value suggests the probability of how much the group means are influencing our target variable. Over here, the variable holiday fails to prove the null hypothesis as this is less than 0.05.

	sum_sq	df	F	PR(>F)
C(yr)	8.281765e+08	1.0	1140.165127	1.645182e-149
C(holiday)	2.982315e+05	1.0	0.410581	5.218829e-01
C(workingday)	1.897657e+07	1.0	26.125377	4.123844e-07
C(mnth)	1.756683e+08	11.0	21.985966	8.868496e-39
C(weekday)	1.621156e+07	6.0	3.719791	1.193190e-03
C(weathersit)	1.856596e+08	2.0	127.800423	4.274469e-48
C(season)	6.632745e+07	3.0	30.438053	1.635986e-18
Residual	5.128140e+08	706.0	NaN	NaN

Fig 5.4 ANOVA Analysis

Now, we need to drop them to remove multicollinearity.
But we have to be sure that we are not losing any information.
So, we tried with dropping multicollinear column and building models and we got below result:

- on dropping weathersit or season we are losing accuracy with a significant amount.
- On dropping holiday we are losing accuracy which is not a significant amount.
- On dropping week_feat(weekday) or working day we are losing accuracy with little amount.

And if we drop holiday, that information is also included in working day.

We just can't be sure by looking at test result and deciding, we need to get all factors. Here two columns most of the time showing collinearity but some datapoints would not be showing collinearity and at that datapoint there could be abrupt difference in bike renting count column which is very important information for our model.

We will drop only holiday column as we have working day column which has information on holiday also, that is why on dropping holiday we are not losing our accuracy. Also in while getting important features, we are getting the last ranking for holiday column with very less importance.

Removed variables:

instant(non important variable)
atemp (colinearity with temp)
registered / casual (colinearity with cnt)
holiday (From both correlation and ANOVA test showing non-importance of this variable, This can account into class imbalance too, so dropping this variable for our further analysis will be fine.)

Chapter 6 Feature Scaling

As discussed, already have numerical variables(temp, hum and windspeed) which are already in normalized form. By normalizing these variables are in the same range, So, we are not required to scale or normalize the data further.

Now further, after removing variables in feature selection and scaling the features to same range now we include some dummy variables for season and weather in python

Chapter 7 Modelling

Now, that our dataset is free from any anomaly, colinearity, and is scaled to the same range. We are ready to make machine learning models to predict bike rental count.

Approach

We'll be using various different machine learning algorithms to make different models. We'll be comparing amongst themselves using different methodologies and after selecting the machine learning model we'll be using hyperparameter tuning to check the best parameters to make model.

before proceeding, let's get familiar with the terms used in our codes.

```
➢ bike_data: dataset containing all columns except which we
removed in EDA
➢ bike_data _wo : its same as bike_data but we removed
outliers from it.
➢ X_train: containing all independent variables (part of
bike_data), used for training model
➢ y_train : containing target variable (part of bike_data),
used for trianinig model
➢ X_test : containing independent variabel (part of
bike_data), used for testing model
➢ y_test: containing target variable(part of bike_data), ,
used for testing model
➢ X_train_wo: containing all independent variables (part of
bike_data_wo), used for training model
➢ y_train_wo : containing target variable (part of
bike_data_wo), used for trianinig model
➢ X_test_wo : containing independent variabel (part of
bike_data_wo), used for testing model
➢ y_test_wo: containing target variable (part of
bike_data_wo), , used for testing model
➢ fit_predict_show_performance: user-defined function,
which will fit our model on training set, and will calculate
and print k-fold (10-fold) cross validation score for
explained_variance , and then will make prediction on
training and test dataset and will print explained_variance
for both training and test dataset.
```

There are two major problems we face while making machine learning models.
>Firstly dividing the data between the training and test set. We want to divide the whole dataset into training and test datasets and also want to maximize each one of them. So, This problem is solved by F fold Cross validation.

K-fold Cross Validation: K-Fold CV is where a given data set is split into a ***K*** number of sections/folds where each fold is used as a testing set at some point. Lets take the scenario of 5-Fold cross validation(K=5). Here, the data set is split into 5 folds. In the first iteration, the first fold is used to test the model and the rest are used to train the model. In the second iteration, 2nd fold is used as the testing set while the rest serve as the training set. This process is repeated until each fold of the 5 folds have been used as the testing set.

So, we would use K-fold cross-validation technique to get the performance of our model.

In our case we'll be using K fold cross validation over MAE(mean absolute error) to find the performance of the machine learning model.

> The second problem we face is choosing the right parameter for model development. Some machine learning algorithms require right parameter for efficient model making, choosing the right parameter is also an important part of the whole process. This problem is countered by Hyperparameter tuning.

GridsearchCV : (Hyperparameter tuning) Hyperparameter are the parameters which we pass an argument to our building function, like kernel, criterion, n_estimators etc. So to get the best values of this gridserchcv is used. In this technique, we make list of these different parameters and then gridsearchcv build a model for every combination of these parameters and then check cross-validation score and based on the score it gives the best combination of hyperparameters. And then we can build our model with the values of hyperparameter given by GridSearchCV. This is called performance tuning and we would use this to tune our model.

7.1 Building models

Now, we'll be making different Models and comparing them amongst them and testing their performance.

Linear Regression:

```
WITH INCLUSION OF OUTLIERS
K-fold MAE
602.0884324188348

training data MAE
586.2328199481575

test data MAE
612.7238949740137


WITHOUT OUTLIERS
K-fold MAE
618.545675628298

training data MAE
599.460692643405

test data MAE
555.3665114236333
```

KNN:

```
WITH INCLUSION OF OUTLIERS
K-fold MAE
736.6488953828172

training data MAE
567.1078767123288

test data MAE
823.9496598639456

WITHOUT OUTLIERS
K-fold MAE
788.8738052026617

training data MAE
606.8000000000001

test data MAE
728.3444444444444
```

SVM

```
WITH INCLUSION OF OUTLIERS
K-fold MAE
1541.633414556555
```

```
training data MAE
1533.4714462802633

test data MAE
1680.3422148463278

WITHOUT OUTLIERS
K-fold MAE
1516.7871963147704

training data MAE
1509.7285185654002

test data MAE
1762.7738668798677
```

Decision Tree Regression

```
WITH INCLUSION OF OUTLIERS
K-fold MAE
661.6183226183518

training data MAE
0.0

test data MAE
703.9387755102041

WITHOUT OUTLIERS
K-fold MAE
666.5878402903811

training data MAE
0.0

test data MAE
589.25
```

Random Forest

```
WITH INCLUSION OF OUTLIERS
K-fold MAE
503.8973115137345

training data MAE
195.1821917808219

test data MAE
488.0448979591837

WITHOUT OUTLIERS
K-fold MAE
480.8524682395644
```

```
training data MAE
203.9949389179756

test data MAE
476.7604166666667
```

XGBegressor

```
WITH INCLUSION OF OUTLIERS
K-fold MAE
470.79039147364193

training data MAE
321.68544842445687

test data MAE
533.6797640015479

WITHOUT OUTLIERS
K-fold MAE
489.300106014891

training data MAE
320.9108204200838

test data MAE
426.6077134874132
```

Hyper-parameter Tuning

Now, we will tune our two best models i.e. Random Forest and XGBRegressor. As we see in previous step, outliers shown by boxplot method in actual was information for our model. So we will tune our model for whole dataset i.e. bike_data. With the help of hyperparameter tuning we would find optimum values for parameter used in function and would increase our accuracy

Random Forest

```
{'max_depth': 16, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 2, 'n_estimators': 600, 'random_state': 1}
```

```
K-fold MAE
473.087908010124

training data MAE
220.11370957058546
```

```
test data MAE
491.48748069718147
```

XGBRegressor Hyperparameter tuning

```
{'gamma': 0, 'learning_rate': 0.055, 'max_depth': 3, 'n_estimators': 250, 'random_state': 1, 'subsample': 0.7}
```

```
K-fold MAE
438.31215437371293

training data MAE
271.81761519549644

test data MAE
481.8166580119101
```

From above result (on tuned parameter), we have the accuracy of our model by a significant amount.

Conclusion

In summary, we firstly performed feature selection and feature engineering to select useful feature and then we then using these features we tried to make machine learning models and compare amongst themselves using different parameters (Hyperparameter tuning) and different sets of training and test data(K fold analysis) and then comparing to each other taking MAE as the error parameter.

Python Code:

```
# In[1]:


import numpy as np
import pandas as pd
import os
import seaborn as sns
import matplotlib.pyplot as plt
plt.style.use('seaborn-whitegrid')
from sklearn.metrics import r2_score
from sklearn.model_selection import cross_val_score


# In[2]:


os.chdir("F:/analytics_basics/bike_prediction")
```

```
# In[3]:


data=pd.read_csv("day.csv")


# In[4]:


data.head()
```

```
# In[5]:


data.dtypes


# In[6]:


data.nunique()


# In[7]:
```

```
cat_cnames = ['season', 'yr', 'mnth', 'holiday', 'weekday',
'workingday', 'weathersit']
num_cnames = ['temp', 'atemp', 'hum', 'windspeed', 'casual',
'registered', 'cnt']


# In[8]:


data[num_cnames].describe()
```

```
# In[9]:


data.isnull().sum()


# In[10]:


fig0, ax0 = plt.subplots(nrows = 1, ncols = 4, figsize=
(15,5))
var = ['temp', 'atemp', 'hum', 'windspeed']
i=0
for j in var:
    ax1 = sns.distplot((data[j]),ax = ax0[i], rug=True,
color="b")
    ax1.set_title('Distribution plot - ' +str(j))
    ax1.set_ylabel('Frequency')
    ax1.set_xlabel(str(j))
    i+=1


# In[11]:


for i in cat_cnames:
    print(i)
    print(data[i].value_counts())
    print("   ")


# In[12]:
```

```
data.groupby('yr')['cnt'].sum().plot.bar(x='yr',y='cnt')


# In[13]:


var2 = ['season', 'mnth', 'weekday', 'weathersit']
j=0
for i in var2:
    df = data.groupby(i)['cnt'].sum()
    df = df.reset_index()
    df.plot.bar(x=i,y='cnt').set_title('Sum plot - '
+str(i))
    gp = data.groupby(by = ['yr', i]).sum().reset_index()
    (sns.catplot(x= i, y = 'cnt', data = gp, col = 'yr',
kind = 'bar'))
    j+=1


# In[14]:


from pandas.tools.plotting import scatter_matrix
scatter_matrix(data[num_cnames], alpha=0.2, figsize=(15,
15), diagonal='kde')


# In[15]:


corr = data[num_cnames].corr()
corr
corr.style.background_gradient()


# In[16]:


# making every combination from cat_cnames
factors_paired = [(i,j) for i in cat_cnames for j in
cat_cnames]
# doing chi-square test for every combination
p_values = []
from scipy.stats import chi2_contingency
for factor in factors_paired:
    if factor[0] != factor[1]:
        chi2, p, dof, ex =
chi2_contingency(pd.crosstab(data[factor[0]],

data[factor[1]]))
        p_values.append(p.round(3))
    else:
        p_values.append('-')
p_values = np.array(p_values).reshape((7,7))
p_values = pd.DataFrame(p_values, index=cat_cnames,
columns=cat_cnames)
print(p_values)


# In[17]:


#ANOVA Analysis
import statsmodels.api as sm
from statsmodels.formula.api import ols


cw_lm=ols('cnt ~ C(yr)+C(holiday)+C(workingday)+
C(mnth)+C(weekday)+ C(weathersit)+C(season)',
```

```
data=data).fit()
print(sm.stats.anova_lm(cw_lm, typ=2))


# In[18]:


# Making dummies
season_dm = pd.get_dummies(data['season'], drop_first=True,
prefix='season')
data = pd.concat([data, season_dm],axis=1)
data = data.drop(columns = ['season'])
weather_dm = pd.get_dummies(data['weathersit'], prefix=
'weather',drop_first=True)
data = pd.concat([data, weather_dm],axis=1)
data = data.drop(columns= ['weathersit'])


# In[19]:


# creating another dataset with removed outliers
# data_wo (data without outliers)- for checking and
comparing the performance with the
data_wo = data.copy()
# dropping outliers from boxplot method
for i in ['windspeed', 'hum']:
    q75, q25 = np.percentile(data_wo.loc[:,i], [75 ,25])
    iqr = q75 - q25
    min = q25 - (iqr*1.5)
    max = q75 + (iqr*1.5)
    data_wo = data_wo.drop(data_wo[data_wo.loc[:,i] <
min].index)
    data_wo = data_wo.drop(data_wo[data_wo.loc[:,i] >
max].index)


# In[20]:


# dropping unwanted columns as decided in feature selection.
data.drop(columns=['instant', 'dteday', 'holiday', 'atemp',
'casual', 'registered'], inplace=True)
data_wo.drop(columns=['instant', 'dteday', 'holiday',
'atemp', 'casual', 'registered'], inplace=True)


data.head()


# In[77]:


####******** Building Machine learning models *******#####


# fuction to check the performance of the regression model
using kfold cross validation on explained variance
# also checking the score with the training and test dataset
from sklearn.metrics import mean_absolute_error
from sklearn.model_selection import cross_val_score
def predictions(regressor_model, X_train, y_train, X_test,
y_test):
    regressor_model.fit(X_train, y_train)
    y_pred1 = regressor_model.predict(X_train)
    y_pred2 = regressor_model.predict(X_test)
    # here we are taking the k fold parameter as 10. It will
divide the whole dataset into 10 equal parts and check
performance taking each part one time as test data and other
```

```
parts as training data
    performance = cross_val_score(estimator=regressor_model,
X = X_train, y = y_train, cv = 10,

scoring='neg_mean_absolute_error')
```

```
k_fold_performance = -(performance.mean())
    print("K-fold MAE")
    print(k_fold_performance)
    print()
    print("training data MAE")
    print(mean_absolute_error(y_pred1, y_train))
    print()
    print("test data MAE")
    print(mean_absolute_error(y_pred2, y_test))


# In[78]:


# splitting dataset in train and test for whole dataset
X = data.drop(columns=['cnt'])
y = data['cnt']
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size = 0.2, random_state = 0)
```

```
# In[79]:


# splitting dataset in train and test for dataset without
outlier
X = data_wo.drop(columns=['cnt'])
y = data_wo['cnt']
from sklearn.model_selection import train_test_split
X_train_wo, X_test_wo, y_train_wo, y_test_wo =
train_test_split(X, y, test_size = 0.2, random_state = 0)


# In[80]:


#   Linear Regression   #
```

```
# building model for dataset
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
print("WITH INCLUSION OF OUTLIERS")


predictions(regressor, X_train, y_train, X_test, y_test)
print()


# building model for dataset without  outliers
regressor = LinearRegression()
print()
print("WITHOUT OUTLIERS")
predictions(regressor, X_train_wo, y_train_wo, X_test_wo,
y_test_wo)


# In[83]:


#         KNN           #
```

```
# building model for dataset bike_data
from sklearn.neighbors import KNeighborsRegressor
regressor = KNeighborsRegressor(n_neighbors=5)
print("WITH INCLUSION OF OUTLIERS")
predictions(regressor, X_train, y_train, X_test, y_test)
print()


# building model for dataset  without outliers
regressor = KNeighborsRegressor(n_neighbors=5)
print("WITHOUT OUTLIERS")
predictions(regressor, X_train_wo, y_train_wo, X_test_wo,
y_test_wo)


# In[84]:


#        SVM            #
```

```
# building model for dataset bike_data
from sklearn.svm import SVR
regressor = SVR()
print("WITH INCLUSION OF OUTLIERS")
predictions(regressor, X_train, y_train, X_test, y_test)
print()


# building model for dataset bike_data_wo i.e. without
outliers
regressor = SVR()
print("WITHOUT OUTLIERS")
predictions(regressor, X_train_wo, y_train_wo, X_test_wo,
y_test_wo)


# In[92]:


# Decision Tree Regression  #


# building model for dataset
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor(random_state=1)
print("WITH INCLUSION OF OUTLIERS")
predictions(regressor, X_train, y_train, X_test, y_test)
print()
```

```
# building model for dataset without outliers
print("WITHOUT OUTLIERS")
predictions(regressor, X_train_wo, y_train_wo, X_test_wo,
y_test_wo)


# In[88]:


#  Random Forest   #


# building model for dataset bike_data
from sklearn.ensemble import RandomForestRegressor
regressor = RandomForestRegressor(random_state=1)
print("WITH INCLUSION OF OUTLIERS")
predictions(regressor, X_train, y_train, X_test, y_test)
print()
```

```
# building model for dataset bike_data_wo i.e. without
outliers
regressor = RandomForestRegressor(random_state=1)
print("WITHOUT OUTLIERS")
predictions(regressor, X_train_wo, y_train_wo, X_test_wo,
y_test_wo)


# In[90]:


#     XGBRegressor      #


# building model for dataset bike_data
from xgboost import XGBRegressor
regressor = XGBRegressor(random_state=1)
print("WITH INCLUSION OF OUTLIERS")
predictions(regressor, X_train, y_train, X_test, y_test)
print()


# building model for dataset bike_data_wo i.e. without
outliers
regressor = XGBRegressor(random_state=1)
print('WITHOUT OUTLIERS')
predictions(regressor, X_train_wo, y_train_wo, X_test_wo,
y_test_wo)


# In[91]:


#        Hyperparameter tuning             #


# tuning Random Forest for dataset #


from sklearn.model_selection import GridSearchCV
# Random Forest hyperparameter tuning
regressor = RandomForestRegressor(random_state=1)
params = [{'n_estimators' : [500, 600, 800],'max_features':
['auto', 'sqrt', 'log2'],
           'min_samples_split':[2,4,6],'max_depth':[12, 14,
16],'min_samples_leaf':[2,3,5],
           'random_state' :[1]}]
grid_search = GridSearchCV(estimator=regressor,
param_grid=params,cv = 5,
                           scoring = 'explained_variance',
n_jobs=-1)
grid_search = grid_search.fit(X_train, y_train)
print(grid_search.best_params_)


# In[97]:


# building Random Forest on tuned parameter
regressor = RandomForestRegressor(random_state=1,
max_depth=16, n_estimators=600,
                                  max_features='auto',
min_samples_leaf=2,min_samples_split=2)
predictions(regressor, X_train, y_train, X_test, y_test)


# In[94]:


# tuning XGBRegressor for dataset   #
```

```
regressor = XGBRegressor(random_state=1)
params = [{'n_estimators' : [250, 300,350, 400,450],
'max_depth':[2, 3, 5],
           'learning_rate':[0.01, 0.045, 0.05, 0.055, 0.1,
0.3],'gamma':[0, 0.001, 0.01, 0.03],
           'subsample':[1, 0.7, 0.8, 0.9],'random_state' :
[1]}]
grid_search = GridSearchCV(estimator=regressor,
param_grid=params,cv = 5,
                           scoring = 'explained_variance',
n_jobs=-1)
grid_search = grid_search.fit(X_train, y_train)
print(grid_search.best_params_)
```

```
# In[100]:


# Building XGBRegressor on tuned parameter
regressor = XGBRegressor(random_state=1,
learning_rate=0.055, max_depth=3, n_estimators=250,
                         gamma = 0, subsample=0.7)
predictions(regressor, X_train, y_train, X_test, y_test)
```