Predicting bike rentals

Pranab Pathak

Contents

1. Introduction.....3
   1.1. Problem Statement.....3
   1.2. Data.....3

2. Methodology.....5
   2.1 Pre-processing.....5
      2.1.1 Outlier Analysis.....5
      2.1.2 Visual data analysis.....5
      2.1.3 Feature engineering and selection.....8
   2.2 Modeling.....11
      2.2.1 Model selection.....11
      2.2.2 Multiple linear regression.....11
      2.2.3 Random forest regression.....12

3. Conclusion.....13
   3.1 Model evaluation.....13
   3.2 Final thoughts about model.....13

Appendix - A
Python Code.....14

Chapter 1: Introduction

1.1 Problem statement

In this case we will try to predict the daily rentals of bikes. Although there hasn't been much of a mention about the goal of the project, based on our intuition we can say that the results obtained through our model can be used to implement better and organised sales promotion during the days on low sales. On the days of high sales the company can use this report to better allocate operational resources. Nonetheless, a good prediction shall come in handy for the organization

1.2 Data

Our prime objective is to build a regression model that will predict the number of bikes rented in a day depending on environmental and seasonal settings. Given below is a sample of the dataset

	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

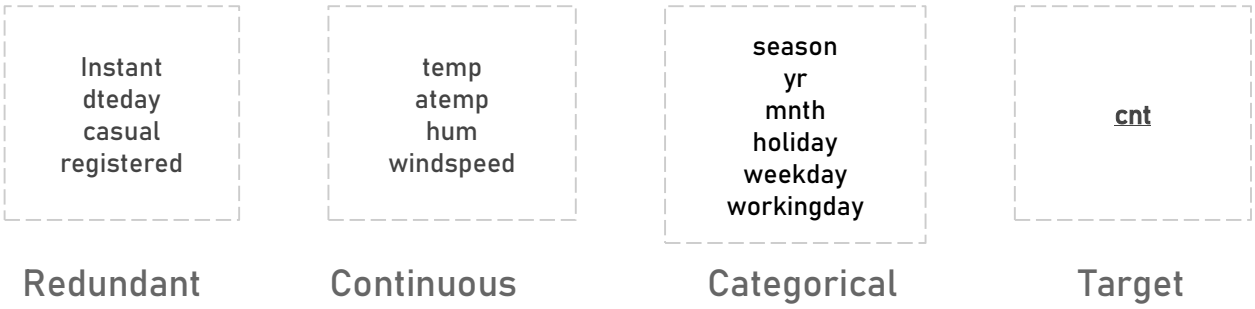
Table 1.1 Bikes rental sample data

The data has a total of 16 columns. The column named 'cnt', the total number of bikes rented, is the target column. The description of the remaining variables are as follows:

- **instant** - A unique sequential ID number for each row
- **dteday** - The date of the rentals
- **season** - The season in which the rentals occurred
- **yr** - The year the rentals occurred
- **mnth** - The month the rentals occurred
- **holiday** - Whether or not the day was a holiday
- **weekday** - The day of the week (as a number, 0 to 7)
- **workingday** - Whether or not the day was a working day
- **weathersit** - The weather (as a categorical variable)
- **temp** - The temperature, on a 0-1 scale
- **atemp** - The adjusted temperature
- **hum** - The humidity, on a 0-1 scale
- **windspeed** - The wind speed, on a 0-1 scale
- **casual** - The number of casual riders (people who hadn't previously signed up with the bike sharing program)
- **registered** - The number of registered riders (people who had already signed up)

Clearly, the variables ‘instant’, ‘casual’ and ‘registered’ are redundant and cannot be used as predictor variables. When instant has unique independent values for each observation, ‘casual’ and ‘registered’ are merely a breakdown of the total rental count.
The variable ‘dtedat’ might be useful in a time series analysis but not in our case. We do not get much information from these above mentioned columns.
The variables ‘temp’, ‘atemp’, ‘hum’, ‘windspeed’ are continuous variables. On the other hand, ‘season’, ‘mnth’, ‘holiday’, ‘weekday’, ‘workingday’, ‘weathersit’ are categorical ones.

Fig 1.1 Variable breakdown



Chapter 2: Methodology

2.1 Pre processing

2.1.1 Outlier analysis

Now that we know what the data is about, we can go ahead with some exploratory data analysis to do some initial investigations on the data. This will help us find patterns and relations between variables. First we will do an outlier analysis on the continuous predictor variables. Outliers are inconsistent data points that might harm our model performance. Removing outliers is an important step in any analysis.
In the figure below the boxplots of the three continuous predictors temperature, humidity and wind-speed are shown.

As can be seen, there are very few outliers for the wind-speed attribute and some inliers for humidity. In this case we use the classic approach, the turkey's method, to removing outliers

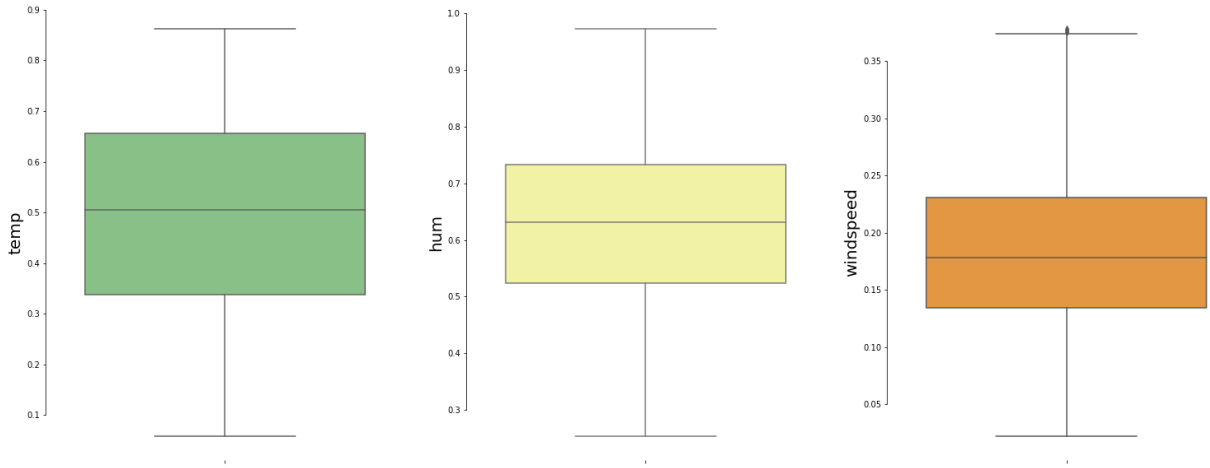


Fig 2.1 Box-plots for continuous variables

2.1.2 Visual analysis

In figure 2.2, we have plotted the probability density functions for the target variable 'cnt'. In fig 2.3 we have done the same for the continuous predictor variables.

The colored line shows the kernel density estimation and the blue line is the normal distribution function. Clearly, the KDEs more or less align with the pdf. This is an indication that there are few outliers in our data

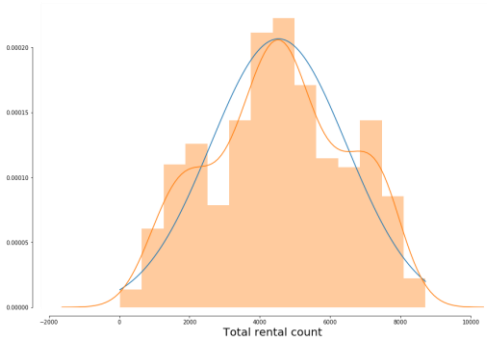


Fig 2.2 density plot for continuous target variable

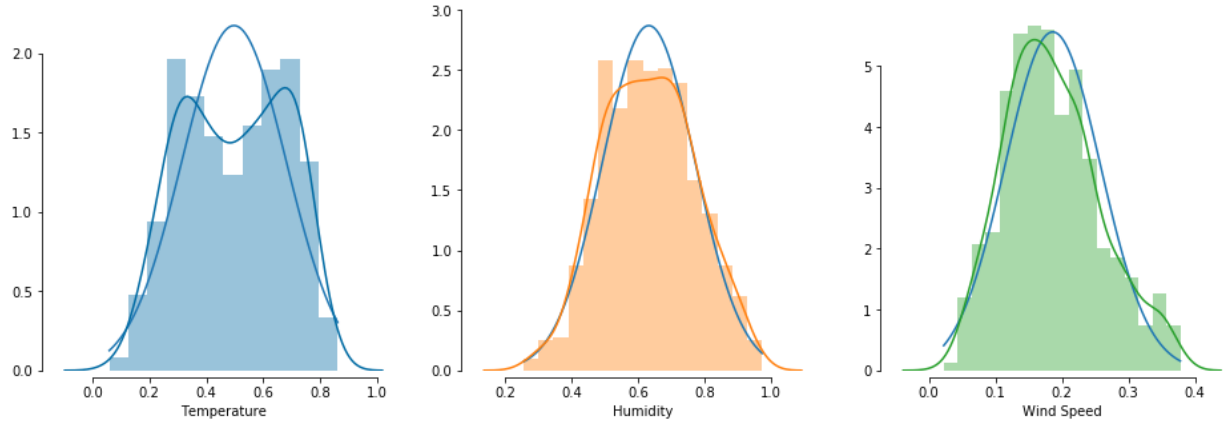


Fig 2.3 density plot for continuous predictor variables

Our continuous predictors look good enough. But we do not know how they relate with the target variable yet. In fig 2.4, we have plotted scatter-plots between the above three predictors and the target variable. As can be seen, there is a good correlation between temperature and the number of bike rentals. The rental count seem to have a positive relation with temperature. On the other hand, there doesn't seem to be much of a dependency on the other two predictors namely humidity and wind-speed

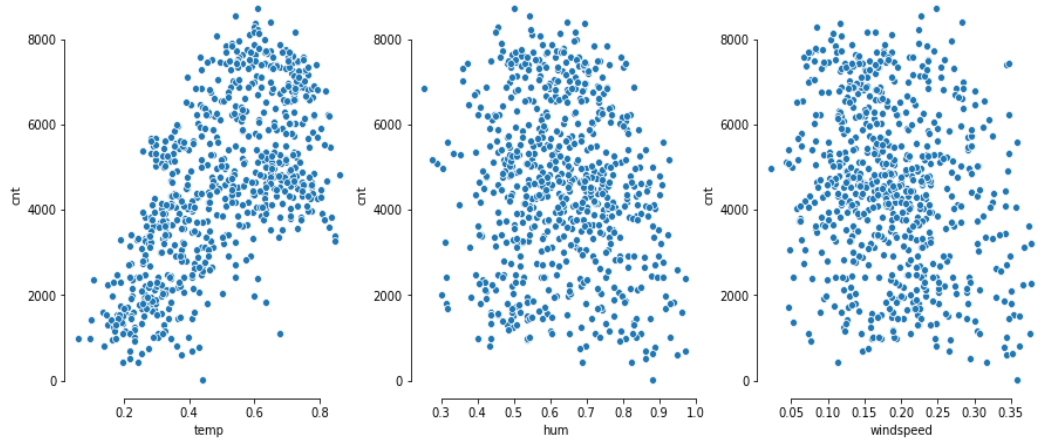


Fig 2.4 Scatter plots for continuous predictor variables

Similar inferences can also be drawn from the heatmap of correlation matrix of the continuous variables as shown in fig 2.5. The heatmap also show us the correlation between the predictors

As we hypothesized, we have a strong correlation of 0.63 between temperature and the rental counts.

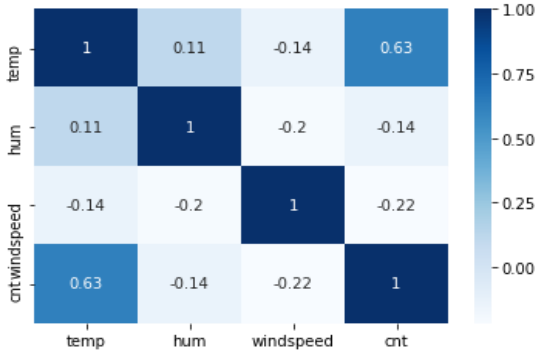


Fig 2.5 Correlation heatmap

Coming to categorical variables now. A good way to see the relationship between a categorical variable and a continuous variable is to see the how the mean value of the later differs for each of the category, also called as anova test. While that is highly numeric, in the world of graphical representations we do that using boxplots. That's exactly what we have done in fig 2.6.

We can see some patterns here.

For the month variable, the rental count keeps on increasing until month 9 and deeps down from there on. This is probably because of seasonal changes. And in support of our hypothesis, the season variable shows a similar pattern.

The weekday and workingday plots doesn't show much of a difference for the respective categories.

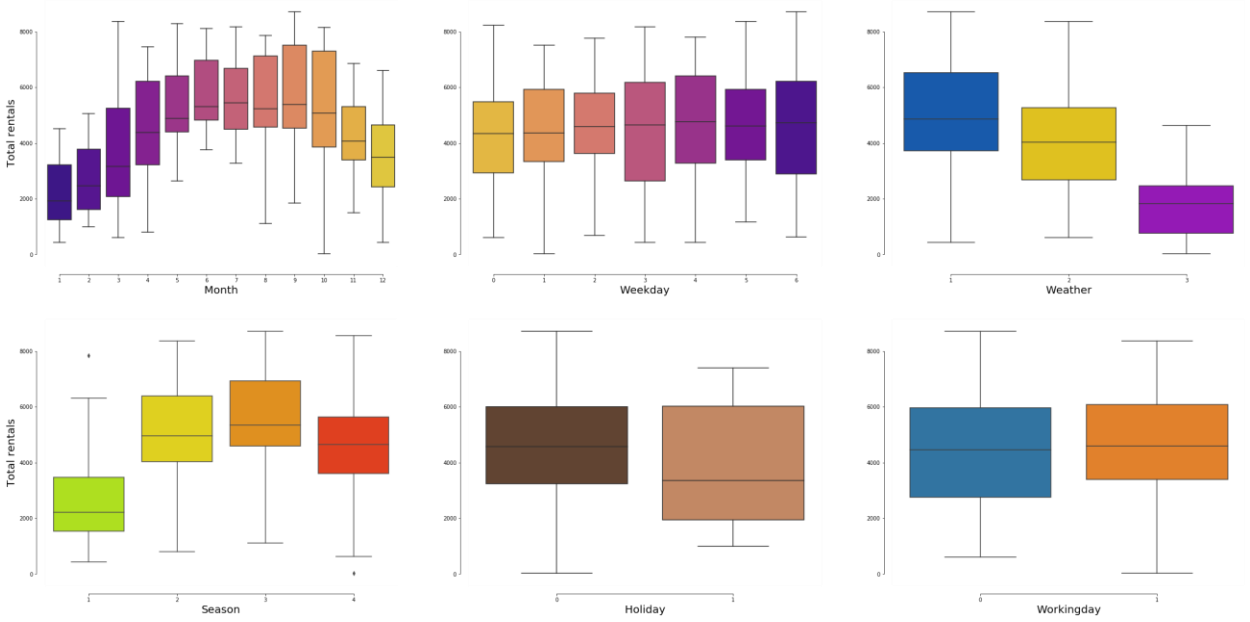


Fig 2.6 Boxplot of the target for each category of each of the categorical variables

We see highest rental count for the weather category 1 and it decreases further. Lastly, the rentals seem to be more in numbers when it is not a holiday.

2.1.3 Feature engineering and selection

We have 6 categorical columns in our data, all of them are *nominal* i.e., there is order associated with them. Since we will be doing regression models, we have made *dummy variables* out of them after which we end up with 29 binary predictor variables. Fig 2.7 give an idea of how the data looks like after this conversion

	yr	holiday	workingday	temp	atemp	hum	windspeed	cnt	mnth_1	mnth_2	mnth_3	mnth_4	mnth_5	mnth_6	mnth_7	mnth_8
0	0	0	0	0.344167	0.363625	0.805833	0.160446	985	1	0	0	0	0	0	0	0
1	0	0	0	0.363478	0.353739	0.696087	0.248539	801	1	0	0	0	0	0	0	0
2	0	0	1	0.196364	0.189405	0.437273	0.248309	1349	1	0	0	0	0	0	0	0
3	0	0	1	0.200000	0.212122	0.590435	0.160296	1562	1	0	0	0	0	0	0	0
4	0	0	1	0.226957	0.229270	0.436957	0.186900	1600	1	0	0	0	0	0	0	0

Fig 2.7(a) Sample data after dummy variable conversion

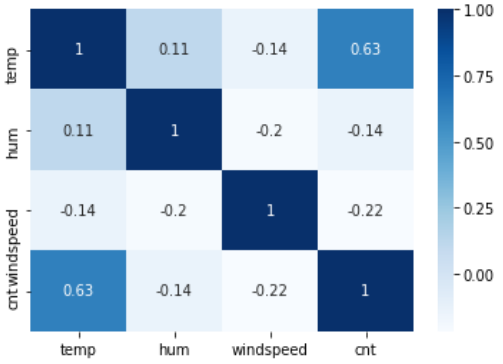
mnth_9	mnth_10	mnth_11	mnth_12	weekday_0	weekday_1	weekday_2	weekday_3	weekday_4	weekday_5	weekday_6	weathersit_1	weathersit_2
0	0	0	0	0	0	0	0	0	0	1	0	1
0	0	0	0	1	0	0	0	0	0	0	0	1
0	0	0	0	0	1	0	0	0	0	0	1	0
0	0	0	0	0	0	1	0	0	0	0	1	0
0	0	0	0	0	0	0	1	0	0	0	1	0

Fig 2.7(b) Sample data after dummy variable conversion

weathersit_3	season_1	season_2	season_3	season_4
0	1	0	0	0
0	1	0	0	0
0	1	0	0	0
0	1	0	0	0
0	1	0	0	0

Fig 2.7(c) Sample data after dummy variable conversion

As we have already seen from the correlation heatmap, with a coefficient of 0.63, the 'temp' variable is the only one that has a significant relation with the target variable 'cnt'. The 'winspeed' column also show some amount of dependency. However, we will only consider variables with a correlation coefficient of 0.25 or higher.



We do the same for our binary features as well. First we checked the correlation between features

and target variable and eliminate the ones with coefficient lower than 0.25 and higher than -0.25. Fig 2.9 below shows the coefficients using a bar plot.

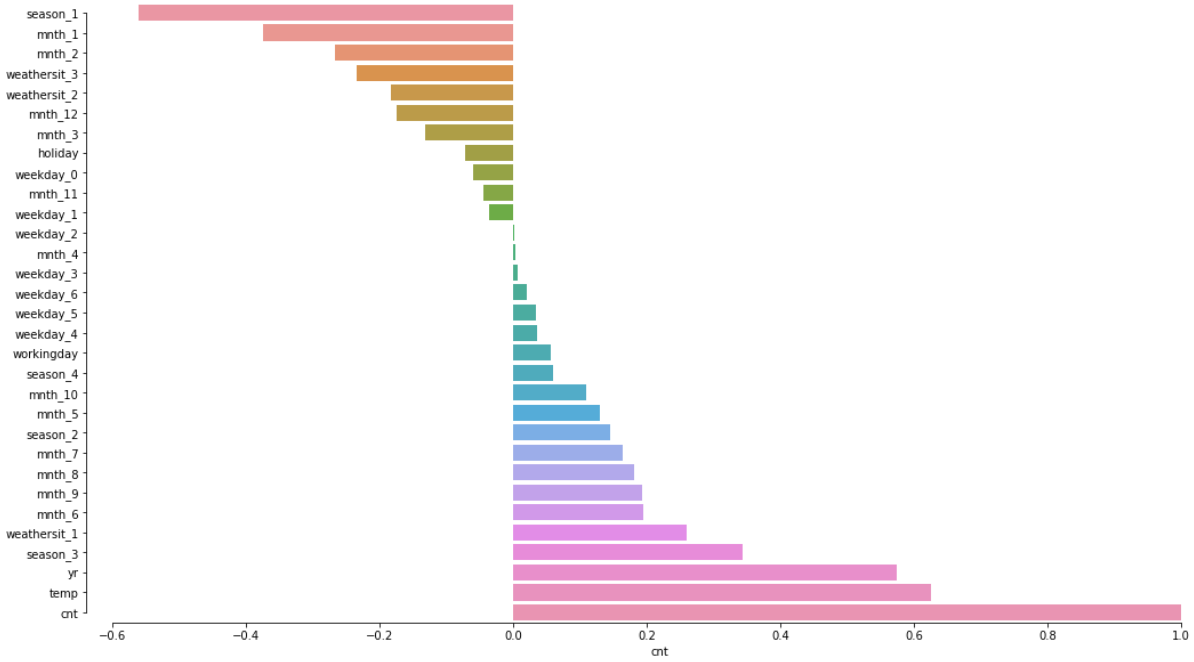


Fig 2.8 Bar plot of correlation coefficients between predictor and target variables

We found the following features to have the highest coefficients

- 'yr' : 0.57
- 'weathersit_1' : 0.26
- 'season_3' : 0.34
- 'temp' : 0.63
- 'mnth_1' : -0.38
- 'mnth_2' : -0.27
- 'season_1' : -0.56

Fig 2.9 shows the heatmap of correlation matrix of the above mentioned features only. We found out that the features *'mnth_1', 'mnth_2', 'season_1', and 'season_3'* have strong correlations between them. This is probably because with changing months we also observe changing seasons. It is very much possible that during month 1 and month 2 we almost always experience season 1. Hence, the month feature can be eliminated.

Also, the 'season_1' and 'season_3' have a correlation of -0.33. Season_1 being better related with the target, we will eliminate season_3.

At the end, we are left with only 4 predictor variables namely *'yr', 'temp', 'weathersit_1' and, 'season_1'*

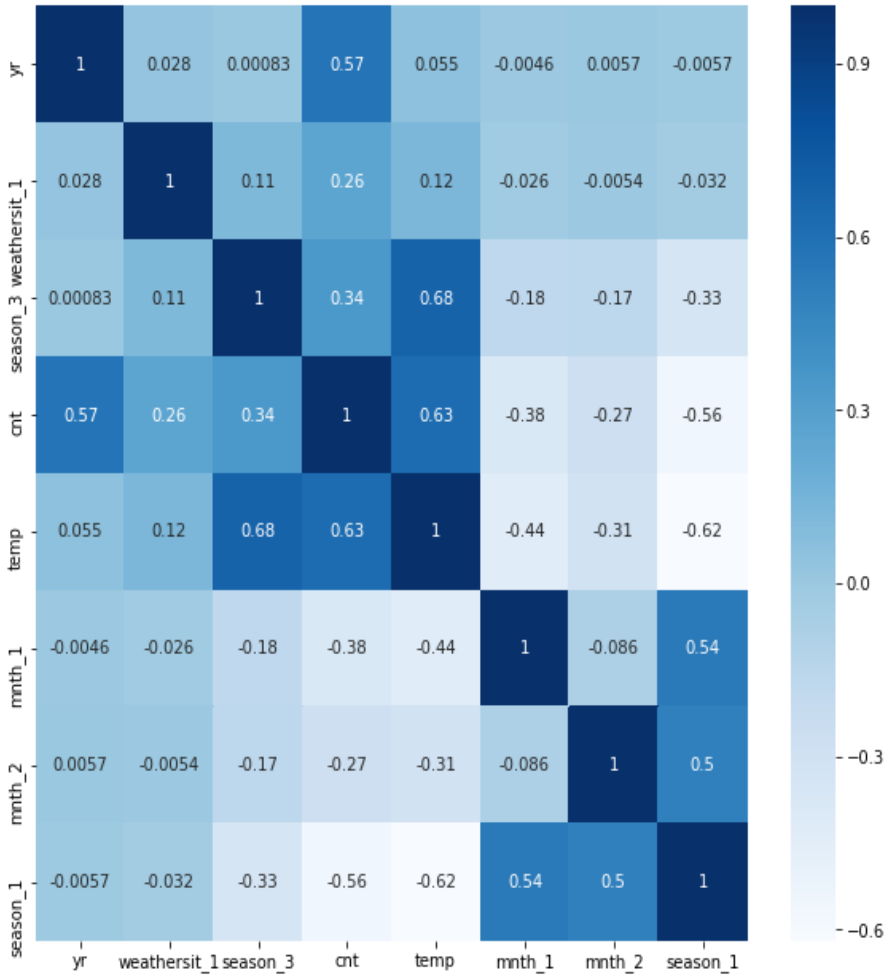


Fig 2.9 Correlation heatmap

Note: We have found that the 'yr' feature is very well correlated with our target variable. This could be because the company expanded pretty well over a year and sales spiked. Although this is an approved feature to predict sales within the years 2011 and 2012, we cannot and must not use it for future predictions for obvious reasons. In this project we are making the test set out of the same data set so, we will keep it in our model.

2.2. Modeling

2.2.1 Model selection

Our target variable is a continuous one. Using a multiple linear regression model makes the most sense. However, we will also try using regression trees method and see how each of the model performs.

2.2.2 Multiple Linear Regression

We did a train-test sampling and used the statsmodel library to get a regression report

```
train = bikes.sample(frac=0.75, axis=0)
test = bikes.drop(train.index, axis=0)
```

```
from statsmodels.api import OLS
OLS(train['cnt'], train[fea_2]).fit().summary()
```

OLS Regression Results

Dep. Variable:	cnt	**R-squared:**	0.962
Model:	OLS	**Adj. R-squared:**	0.962
Method:	Least Squares	**F-statistic:**	2726.
Date:	Sat, 20 Apr 2019	**Prob (F-statistic):**	0.00
Time:	13:34:41	**Log-Likelihood:**	-4458.7
No. Observations:	538	**AIC:**	8927.
Df Residuals:	533	**BIC:**	8949.
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
yr	2159.7961	83.015	26.017	0.000	1996.719	2322.873
temp	6528.1868	171.210	38.130	0.000	6191.857	6864.517
weathersit_1	879.8015	85.861	10.247	0.000	711.134	1048.469
season_1	-900.1240	96.317	-9.345	0.000	-1089.331	-710.917
season_3	-725.3524	116.853	-6.207	0.000	-954.901	-495.804

Omnibus:	76.956	**Durbin-Watson:**	2.062
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	154.976
Skew:	-0.810	**Prob(JB):**	2.23e-34
Kurtosis:	5.071	**Cond. No.**	5.25

As it is seen from the *Adjusted R-squared value,* we can explain about 96% of the variance. This is pretty impressive.
Also, looking at the P values and F-statistic, we can reject the null hypothesis that our target variable does not depend on these features.

2.2.3 Random forest regressor

A Random Forest is an ensemble technique capable of performing both regression and classification tasks with the use of multiple decision trees and a technique called

Bootstrap Aggregation, commonly known as bagging. What is bagging you may ask? Bagging, in the Random Forest method, involves training each decision tree on a different data sample where sampling is done with replacement.
The basic idea behind this is to combine multiple decision trees in determining the final output rather than relying on individual decision trees. If you want to read more on Random Forests, I have included some reference links which provide in depth explanations on this topic.

Chapter 3 Conclusion

3.1 Model Evaluation

Now that we have a few models for predicting the target variable, we need to decide which one to choose. There are several criteria that exist for evaluating and comparing models. We can compare the models using any of the following criteria:

1. Predictive Performance
2. Interpretability
3. Computational Efficiency

In our case we will use the root mean squared error value, which falls under the category of predictive performance

```
train = bikes.sample(frac=0.75, axis=0)
test = bikes.drop(train.index, axis=0)

from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(train[fea_2], train['cnt'])
predictions = lr.predict(test[fea_2])

from sklearn.metrics import mean_squared_error
RMSE = mean_squared_error(test['cnt'], predictions)**0.5
print(RMSE)
```

```
814.6680308696821
```

On running the above model several times we found that the RMSE value always lies between 800 and 1000. If we see the range of our target variable ‘cnt’ , the range is around 9000.

The RMSE value is around 10% of this range. We can say that the model performs well with our data

Let’s run a random forest regressor model and see how it performs

```
train = bikes.sample(frac=0.75, axis=0)
test = bikes.drop(train.index, axis=0)

from sklearn.ensemble import RandomForestRegressor
rfr = RandomForestRegressor()
rfr.fit(train[fea_2], train['cnt'])
predictions = rfr.predict(test[fea_2])

from sklearn.metrics import mean_squared_error
RMSE = mean_squared_error(test['cnt'], predictions)**0.5
print(RMSE)
```

```
847.4553966683849
```

3.2 Final thought about the models

We can see that both models perform well and therefore we can select either of the two models without any loss of information.

Appendix - A

Python Code

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns

bikes = pd.read_csv('day.csv')
bikes.head()

bikes.describe()

```

Fig 2.1

```
fig = plt.figure(figsize=(26,10))
pal = ['Accent','Accent_r','YlOrBr']
for i, f in enumerate(['temp','hum','windspeed']):
    fig.add_subplot(1,3,i+1)
    plt.ylabel(f, fontsize=20)
    sns.boxplot(y=f, data=bikes, palette=pal[i])
    sns.despine(trim=True, offset=5)

for i in ['hum', 'windspeed']:
    p25, p75 = np.percentile(bikes[i], [25,75])
    iqr = p75-p25
    max = p75 + 1.5*iqr
    min = p25 - 1.5*iqr
    bikes = bikes.drop(bikes[bikes[i]>max].index, axis=0)
    bikes = bikes.drop(bikes[bikes[i]<min].index, axis=0)

```

Fig 2.2

```
plt.figure(figsize=(15,10))

from scipy.stats import norm
x = np.linspace(22,8714,8000)
y = norm.pdf(x, loc = np.mean(bikes['cnt']), scale = np.std(bikes['cnt']))
plt.plot(x,y)

ax = sns.distplot(bikes['cnt'], hist=True)
ax.set_xlabel('Total rental count', fontsize=20)
sns.despine(trim=True, offset=15)

```

Fig 2.3

```
fig = plt.figure(figsize=(16,5))
cb_dark_blue = (0/255,107/255,164/255)
cb_orange = (255/255, 128/255, 14/255)
cb_green = (40/255, 160/255, 44/255)
pal = [cb_dark_blue,cb_orange,cb_green]
labels = ['Temperature', 'Humidity', 'Wind Speed']
for i, f in enumerate(['temp','hum','windspeed']):
    fig.add_subplot(1,3,i+1)

    x = np.linspace(np.min(bikes[f]), np.max(bikes[f]), 100)
    y = norm.pdf(x, loc=np.mean(bikes[f]), scale=np.std(bikes[f]))
    plt.plot(x,y)

    ax = sns.distplot(bikes[f], color=pal[i])
    sns.despine(trim=True, offset=5)
    ax.set_xlabel(labels[i])

```

Fig 2.6

```
fig = plt.figure(figsize=(40,20))
pal = ['plasma','plasma_r','prism', 'prism_r', 'copper', 'tab10']
xlabels = ['Month', 'Weekday', 'Weather', 'Season', 'Holiday', 'Workingday']
for i, f in enumerate(['mnth','weekday','weathersit', 'season', 'holiday',
'workingday']):
    fig.add_subplot(2,3,i+1)
    ax = sns.boxplot(x=f, y='cnt', data=bikes, palette=pal[i])
    plt.xlabel(xlabels[i], fontsize=20)
    if i==0 or i==3:
        ax.set_ylabel('Total rentals', fontsize=20)
    else:
        ax.set_ylabel('')
    sns.despine(trim=True, offset=15)

```

Fig 2.4

```
fig = plt.figure(figsize=(15,6))

for i,f in enumerate(['temp','hum','windspeed']):
    fig.add_subplot(1,3,i+1)
    ax = sns.scatterplot(f, 'cnt', data=bikes)
    sns.despine(trim=True)

cont_vars = ['temp', 'hum', 'windspeed','cnt']

```

Fig 2.5

```
fig4 = plt.figure(figsize=(6,4))
sns.heatmap(bikes[cont_vars].corr(), cmap='Blues', annot=True)
plt.show()

for value in ['mnth','weekday','weathersit','season']:
    temp = pd.get_dummies(bikes[value], prefix = value)
    bikes = pd.concat([bikes, temp], axis=1)
    del bikes[value]

cat_vars = ['yr', 'holiday', 'workingday','mnth_1', 'mnth_2',
       'mnth_3', 'mnth_4', 'mnth_5', 'mnth_6', 'mnth_7', 'mnth_8', 'mnth_9',
       'mnth_10', 'mnth_11', 'mnth_12', 'weekday_0', 'weekday_1', 'weekday_2',
       'weekday_3', 'weekday_4', 'weekday_5', 'weekday_6', 'weathersit_1',
       'weathersit_2', 'weathersit_3', 'season_1', 'season_2', 'season_3',
       'season_4']

```

Fig 2.8

```
plt.figure(figsize=(18,10))
sns.barplot(bikes[cat_vars+ ['cnt', 'temp']].corr()['cnt'].sort_values(),
bikes[cat_vars+ ['cnt','temp']].corr()['cnt'].sort_values().index)
sns.despine(trim=True)

g = bikes[cat_vars+ ['cnt', 'temp']].corr()['cnt']
g_fea = list(g[g>0.25].index) + list(g[g<-0.25].index)


```

Fig 2.9

```
fig4 = plt.figure(figsize=(10,10))
sns.heatmap(bikes[g_fea].corr(), cmap='Blues', annot=True)
plt.show()

fea_2 = ['yr', 'temp','weathersit_1','season_1']

train = bikes.sample(frac=0.75, axis=0)
test = bikes.drop(train.index, axis=0)

from statsmodels.api import OLS
OLS(train['cnt'], train[fea_2]).fit().summary()

train = bikes.sample(frac=0.75, axis=0)
test = bikes.drop(train.index, axis=0)

from sklearn.linear_model import LinearRegression
```

```
lr = LinearRegression()
lr.fit(train[fea_2], train['cnt'])
predictions = lr.predict(test[fea_2])

from sklearn.metrics import mean_squared_error
RMSE = mean_squared_error(test['cnt'], predictions)**0.5
print(RMSE)

train = bikes.sample(frac=0.75, axis=0)
test = bikes.drop(train.index, axis=0)

from sklearn.ensemble import RandomForestRegressor
rfr = RandomForestRegressor()
rfr.fit(train[fea_2], train['cnt'])
predictions = rfr.predict(test[fea_2])

from sklearn.metrics import mean_squared_error
RMSE = mean_squared_error(test['cnt'], predictions)**0.5
print(RMSE)

train = bikes.sample(frac=0.75, axis=0)
test = bikes.drop(train.index, axis=0)

from sklearn.ensemble import GradientBoostingRegressor
gbr = GradientBoostingRegressor()
gbr.fit(train[fea_2], train['cnt'])
predictions = gbr.predict(test[fea_2])

from sklearn.metrics import mean_squared_error
RMSE = mean_squared_error(test['cnt'], predictions)**0.5
print(RMSE)
```