Predicting Bike Rental Count

Vipin Kumar

25 October 2016

Contents

Chapter 1

Introduction

1.1 Problem Statement

The objective of this Case is to Predication of bike rental count on daily based on the environmental and seasonal settings.

1.2 Data

Our task is to build regression models which will predict the count of bikes rented on the basis of environmental and seasonal settings. Given below is a sample of the data set that we are using to predict the count:

Table 1.1: Bike Rental Count Sample Data (Columns: 1-8)

instant	dteday	season	yr	mnth	holiday	weekday	workingday
1	01-01-2011	1	0	1	0	6	0
2	02-01-2011	1	0	1	0	0	0
3	03-01-2011	1	0	1	0	1	1
4	04-01-2011	1	0	1	0	2	1
5	05-01-2011	1	0	1	0	3	1

Table 1.2: Bike Rental Count Sample Data (Columns: 8-16)

weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
2	0.344167	0.363625	0.805833	0.160446	331	654	985
2	0.363478	0.353739	0.696087	0.248539	131	670	801
1	0.196364	0.189405	0.437273	0.248309	120	1229	1349
1	0.2	0.212122	0.590435	0.160296	108	1454	1562
1	0.226957	0.22927	0.436957	0.1869	82	1518	1600

As you can see in the table above we have15 variables, using which we have to correctly predict the 16^{th} variable which is count of rented bikes (“cnt”).

Chapter 2

Methodology

2.1 Pre-Processing

Any predictive modelling requires that we look at the data before we start modelling. However, in data mining terms *looking at data* refers to so much more than just looking. Looking at data refers to exploring the data, cleaning the data as well as visualizing the data through graphs and plots. This is often called as **Exploratory Data Analysis**.
We'll first have look at the structure of data and convert the categorical variables which are numeric in given table into categorical data. Now, from general understanding we can conclude that attribute 'instant'
have no effect on the count variable as it's merely an index. Also, 'dteday' attribute which is the date attribute is of no use, as we are not doing time series analysis here. The attributes 'casual' and 'registered' are just division of our target count variable and as we are just interested in the total count we can exclude them as well from our analysis. The categorical variable 'yr' with two categories each corresponding to an year is also not relevant to this problem as we are trying to predict the expected count of future i.e. next year and so on. But the category of 'yr' variable which may be added for next years would have no trace in our model as there is no previous data available for that category.
We'll exclude variables 'instant', 'dteday', 'casual', 'registered' and 'yr' due to above reasons.
Now we have a dataframe of 11 variables.
To start the pre-processing, we will first try and look at all the probability distributions of the continuous variables. Most analysis like regression, require the data to be normally distributed. We can visualize that in a glance by looking at the probability distributions or probability density functions of the variable.

In Figure 2.1 we have plotted the probability density functions of all the continuous variables available in the data. The blue lines represent the normal distribution.

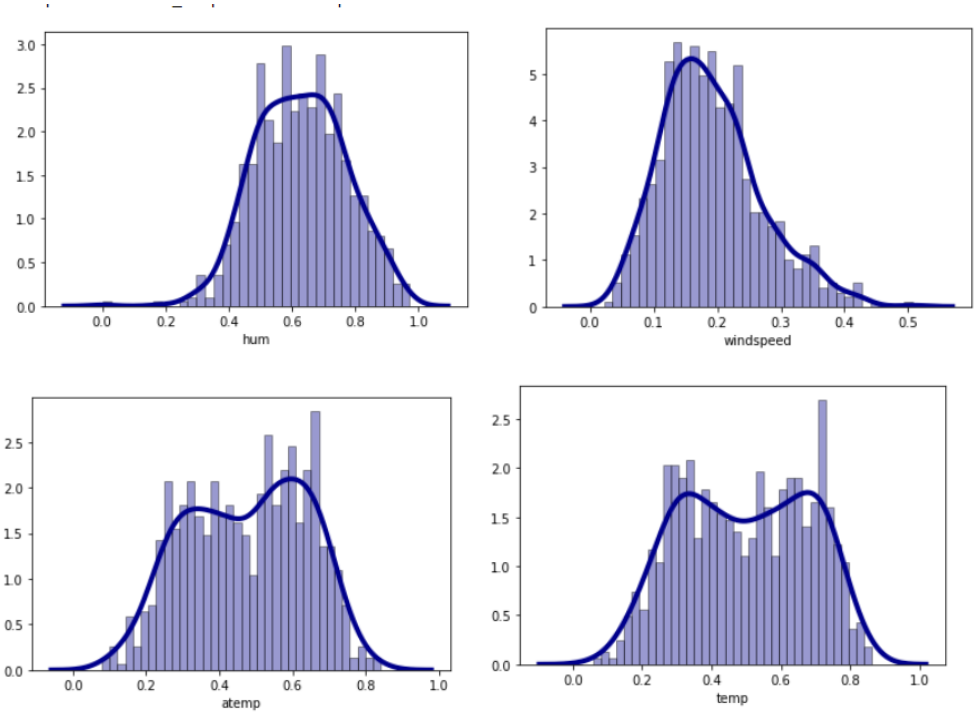

Fig 2.1: Probability Distribution of continuous variables

2.1.1 Missing Value Analysis

It includes analysing the data for any missing value present. Missing values can make our predictive modelling difficult at a later point of time. So, it's better to remove or impute the missing values beforehand.

Our check for the missing values on the given data gave us zero output. Therefore, we conclude that there is no missing value in our data.

2.1.2 Outlier Analysis

We can clearly observe from these probability distributions that most of the variables are skewed. The skew in these distributions can be most likely explained by the presence of outliers and extreme values in the data.

We visualize the outliers using box plot given in Fig 2.2

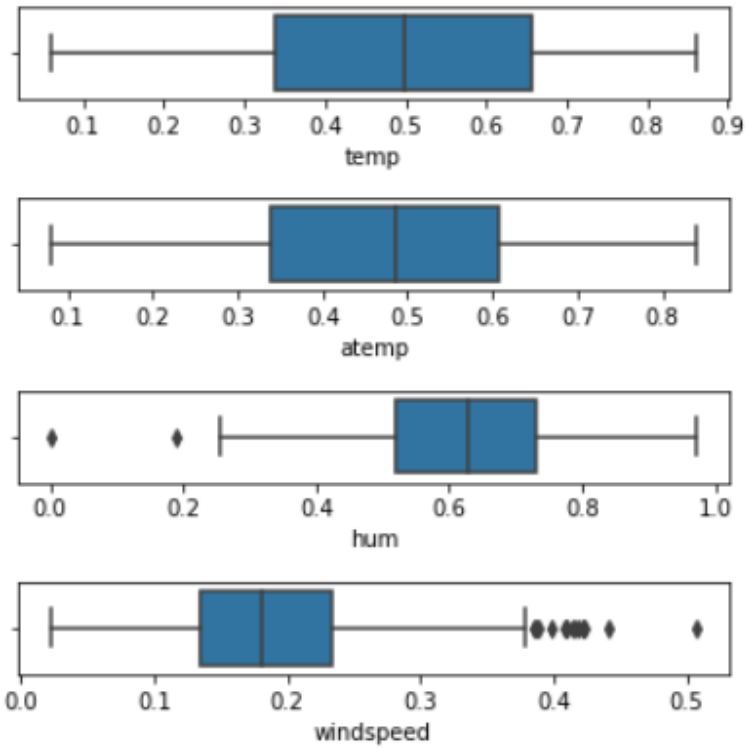

Fig. 2.2 BOX PLOT

We can clearly see from the box plot that the variable 'hum' and 'windspeed' have some outliers which can degrade our model performance. Hence, we'll remove these outliers from the data.

After removal of the outliers our data comes down to a shape of (717,11).

2.1.3 Correlation Analysis

Multicollinearity refers to a situation in which two or more explanatory variables in a multiple regression model are highly linearly related. We have perfect multicollinearity if, the correlation between two independent variables is equal to 1 or −1. Multicollinearity adds redundant data while modelling which ends with noise in the model and affects our prediction adversely.

Therefore, correlation analysis is done to detect multicollinearity and if found above a threshold the variable is removed.

After our analysis we found that the correlation between 'temp' and 'atemp' variable is more than 0.99 i.e. they have 99% similar information to give to our model. Hence, we'll remove one of these variables, 'atemp' variable is removed in this case. Fig 2.2 shows the red colour between temp and atemp variables which corresponds to collinearity 1.

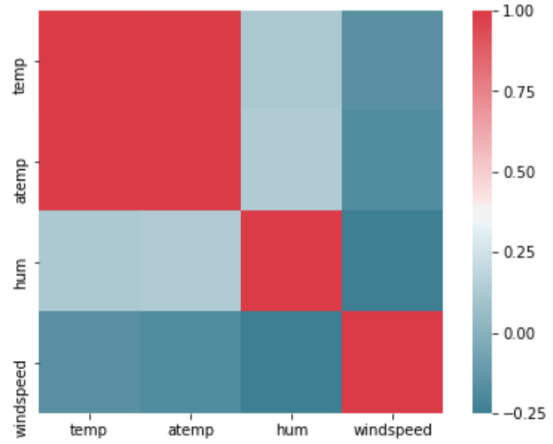

Fig. 2.3 Correlation Plot

2.1.4 Chi square test

Where the correlation analysis is used to extract the relation between continuous variables, chi square test is used to extract the dependencies of categorical variables. In the given data, we have six categorical variables and we'll analyse them for the null hypothesis that is 'variables are independent'. If the chi square test for any variable give p value more than 0.9 which is our threshold, for any variable than that variable will be excluded from modelling. Chi-square test gave p values of all the categorical variables below 0.7. Hence, the null hypothesis is accepted that the variables are independent.

2.1.5 Normalisation

Normalisation is a part of data scaling. Data scaling is used to minimize the biasing created in the model by attributes of high magnitude over the low magnitude attributes. Standardization can also be used as a scaling method, but it requires data to be in normal. And we have seen from our distribution plots that the attributes we have are somewhat skewed in nature.

Although we don't have much difference in the magnitudes of our data but still normalization is performed to further enhance the quality of data.

2.2 Modelling

After completing all the pre-processing, we have a better cleaner data which will produce less noise and better models. As we can clearly see that this a regression problem, we'll start with the basic simple algorithm for regression modelling that is multiple linear regression and then we'll try to migrate to bit complex models to try and improve the accuracy of model.
Data is first split into train data (Used to train the model) and test data (used to test the model with unseen data).
In this case we are dividing the data in the ratio of 80% (train) and 20% (test), however we can use different ratios to make this split.

2.2.1 Multiple linear regression

model1 = lm(cnt ~., data = training)
summary(model1)

```
 Call:
lm(formula = cnt ~ ., data = training)

Residuals:
    Min      1Q  Median      3Q     Max
-3452.2  -983.4  -261.7  1056.2  3140.4

Coefficients: (1 not defined because of singularities)
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  2769.66     340.17   8.142 2.65e-15 ***
season2       937.11     336.13   2.788  0.00549 **
season3       749.54     404.49   1.853  0.06441 .
season4      1585.93     335.61   4.725 2.92e-06 ***
mnth2         130.38     272.13   0.479  0.63206
mnth3          48.60     316.80   0.153  0.87812
mnth4         110.27     471.19   0.234  0.81505
mnth5         212.98     498.51   0.427  0.66937
mnth6        -115.73     524.63  -0.221  0.82549
mnth7        -842.90     588.28  -1.433  0.15248
mnth8        -173.11     563.66  -0.307  0.75886
mnth9         501.26     492.75   1.017  0.30948
mnth10        232.89     451.30   0.516  0.60604
mnth11       -278.61     426.63  -0.653  0.51399
mnth12        -17.19     342.10  -0.050  0.95995
holiday1     -882.20     323.77  -2.725  0.00664 **
weekday1      223.54     205.41   1.088  0.27696
weekday2      179.74     199.91   0.899  0.36900
weekday3      204.93     206.11   0.994  0.32053
weekday4       86.89     200.51   0.433  0.66493
weekday5      297.85     202.45   1.471  0.14180
weekday6      338.94     201.35   1.683  0.09288 .
workingday1       NA         NA      NA       NA
weathersit2  -207.44     146.90  -1.412  0.15849
weathersit3 -1796.44     393.65  -4.564 6.22e-06 ***
temp         5013.43     619.81   8.089 3.92e-15 ***
hum         -2490.40     412.50  -6.037 2.90e-09 ***
windspeed   -1180.28     294.89  -4.002 7.13e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1269 on 546 degrees of freedom
Multiple R-squared:  0.5879,    Adjusted R-squared:  0.5682
```

F-statistic: 29.95 on 26 and 546 DF, p-value: < 2.2e-16

As you can see the Adjusted R-squared value, we can explain only about 56% of the data using our multiple linear regression model. Using significance codes, we identify that the attributes which contribute most to the model are 'temp', 'hum', 'windspeed', 'weathersit', 'season' and 'holiday'.

Lasso and Ridge regressions are also tested but didn't enhance the model accuracy. No overfitting present in our model can be the reason for that.

PREDICTION AND MAPE

The mean absolute percentage error (MAPE), also known as mean absolute percentage deviation (MAPD), is a measure of prediction accuracy of a forecasting method in statistics, for example in trend estimation, also used as a loss function for regression problems in machine learning.

We'll predict the cnt variable for test data and compare it with the actual count given to calculate MAPE.

```
MAPE = function(y, yhat){
  mean(abs((y - yhat)/y))*100
}
predictions_LR = predict(model1, testing[,1:9])
MAPE(testing[,10], predictions_LR)
```

ERROR RATE : 27.86 %

2.2.2 Support Vector Regressor

SVR is used with three algorithms here which are eps-regression, nu-regression and BoundConstrain SVM. All the algorithms are further used with different kernels available for each algorithm.

The eps-regression with radial kerenel gave the least errors.

```
fitepsrad<-svm(cnt~., data=training,type="eps-regression",kernel="radial",cross=573)
summary(fitepsrad)
```

```
Call:
svm(formula = cnt ~ ., data = training, type = "eps-regression", kernel = "radial", epsilon = 0.6, cross = 573
)

Parameters:
   SVM-Type:  eps-regression
 SVM-Kernel:  radial
       cost:  1
      gamma:  0.03571429
    epsilon:  0.6

Number of Support Vectors:  237

573-fold cross-validation on training data:

Total Mean Squared Error: 1445166
Squared Correlation Coefficient: 0.6170122
```

PREDICTION AND MAPE

```
predepsrad<-predict(fitepsrad,testing[,1:9])
MAPE (testing[,10], predepsrad)
```

ERROR RATE: 24.81 %

2.2.2 K Nearest Neighbour

KNN is tested for different values of K value

```
for( j in 1:15){
 KNN_model = knnreg(training[,1:9],training$cnt, k = j)
 KNN_Predictions= predict(KNN_model,testing[,1:9])
 print(MAPE(testing[,10],KNN_Predictions))
 }
```

You can see the change in error with the change in k value in Fig 2.4.

X-axis represents the MAPE and Y-axis represents the K-value.

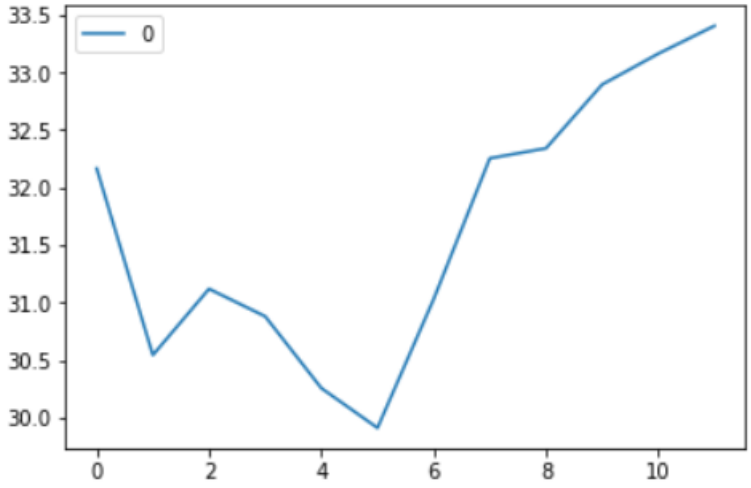

Fig 2.4: K vs MAPE plot

As we can see MAPE touches the least value of nearly 30% when K is 5

ERROR RATE: 30.00 %

Chapter 3

Conclusion

3.1 Model Evaluation

Broadly, three methods are used in this project to model our predictor. After a fine pre-processing of data all of them gave an error percentage of 30 or less. MAPE of these algorithms are stated below:

Multiple Linear Regression: 27.86 %
SVR (eps) : 24.81 %
KNN : 30.00 %

Clearly, SVR serves as the best method with least accuracy of 24.81%

Now, we can further tune the parameters of our best model to decrease its error further

```
tune1=tune(svm,cnt~. , data=training, type="eps-regression",kernel="radial",ranges =
list(cost=c(1.0,1.1,1.2,1.3,1.4,1.5,1.6,1.7,1.8,1.9)))
summary(tune1)
#BestTune cost=1.6

tune2=tune(svm,cnt~. , data=training, type="eps-regression",kernel="radial",ranges =
list(gamma=c(0.001,0.01,0.1,0.2,0.3,0.4,0.5)))
summary(tune2)
#best tune gamma=0.1(default value)

tune5=tune(svm,cnt~. , data=training, type="eps-regression",kernel="radial",ranges =
list(epsilon=c(0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9)))
summary(tune5)

#best tune epsilon=0.6

fitepsradtune<-svm(cnt~., data=training,type="eps-
regression",kernel="radial",cost=1.6,gamma=0.1,epsilon=0.6,cross=573)
summary(fitepsradtune)

# Total Mean Squared Error: 1372250
#Squared Correlation Coefficient: 0.6370695

predepsradtune<-predict(fitepsradtune,testing[,1:9])
MAPE(testing[,10],predepsradtune)

#Final Error :6.86
```

3.2 Model Selection

The best tuned model gives more error then our model before tuning.
Hence, we conclude that SVR eps-regression with radial kernel and other default parameters [cost: 1, ga mma: 0.03571429, epsilon: 0.6] is our bets model with an error rate of 24.8% and accuracy of 75.2%