Predicting Daily Bicycle Demand for Capital Bikeshare

Team Ensemble

EBAC 4 (Mixed Group), Institute of Systems Science, NUS

[Kriti: A0163206N,Muni: A0163382E,Pooja: A0163281J,Pradeep: A0163453H,Sambit: A0163285B]

Abstract

This work is part of assignment to apply Neural Network and other Individual and Ensemble methods on a daily bike Share data provided by Capital Bikeshare company. The further goal is to compare the models and predict which will work better for the company to avoid losses due to over or under stocking. Bike Sharing concept has become very popular recently in metropolitan cities like Shanghai, Singapore, Washington D.C, where infrastructure, lifestyle and other attributes promotes usage of bicycles to live a healthy lifestyle and save time in some cases where otherwise car travel is not easy during peak office hours. In this paper, we analyzed and propose usage of our Ensemble Stacking method which performed better than other Individual models tried across different scenarios. Error chart over time shows a low shelf-life for these models hence continuous training of the data is important factor.

1 EXECUTIVE SUMMARY

Our business problem is to predict total bicycle demand for the next day using different available input parameters like Day, Weather and past Demand information. Day information for the next day is known beforehand, however Weather information is known only up to the current day. Furthermore, past demand information is available only till the previous day since total demand for current day is not known before day end (4pm is the cut-off time).

Additionally, our business goal revolves around profit maximization where loss due to over or under prediction is asymmetric ($2 for each unit of over-prediction vs. $1 for under-prediction). We used following models to forecast the demand for next day:

- A combination of ARIMA based time series forecasting,
- Decision Tree Regression
- Neural Net
- GLM
- Random Forest (Bagging)
- Ensemble method (Stacking)

Some models predictions provide higher profit than the Naïve benchmark (yesterday's actual) model for the test period (2012).
While benchmark model gives $1.443mn total profit (35.2% ROI), our best performing single model GLM results into a total profit of $1.559mn (39% ROI). The final Ensemble, created using a stacking method increased the total profit further to $1.582mn (40% ROI). Single Neural network didn't perform well due to their inherent overfitting nature.

All models show some deterioration in Q4, 2012. Prediction performance goes down after re-weighing the parameters based on 1.5 years of data. We attribute this to imbalance in data as models learn more about Jan-Jun features than Jul-Dec and we test the models for Jul-Dec period only. Data balancing helps when we re-weigh each model using a weight function (1 for Jan-Jun days and 2 for Jul-Dec days). We, however, feel that the models should be rebuilt (not simply parameters re-weighted using same predictors) at least on a half-yearly basis to cater to the growing and dynamic business environment of this industry.

All the profit calculations over different models were done using the following formula:

Profit *= min(Actualt , Predt)*3 - Predt*2 where t = 1, 2, 3…..,366*

1.1 DATA SELECTION AND PREPROCESSING

Upon analyzing the dataset, we can categorize the data in following segments:

- **Days** variables**:** These variables are day of week, weekend, holiday, season. These variables are known in advance for predicting the demand and we feel that there is requirement to create a lag variable for them.
- **Weather** variables: Variables like temperature, humidity, wind-speed could be segmented in to this category
- **Demand** variables: Variables like casual, Registered, cnt comes under this category. These variables are known at the end of the day.

Raw daily dataset contains 16 columns time series for 2011 and 2012. It contains total 731 records. Three variables (instant, dteday, and yr: year) could be ignored as they are simple index variable. yr variable simply divides the data in to 2 years.

R summary output of the various variable is as shown below:

```
'data.frame':          731 obs. of  15 variables:
$ season    : int  1 1 1 1 1 1 1 1 1 1 ...
$ mnth      : int  1 1 1 1 1 1 1 1 1 1 ...
$ holiday   : int  0 0 0 0 0 0 0 0 0 0 ...
$ weekday   : int  6 0 1 2 3 4 5 6 0 1 ...
$ workingday: int  0 0 1 1 1 1 1 0 0 1 ...
$ weathersit: int  2 2 1 1 1 1 2 2 1 1 ...
$ temp      : num  0.344 0.363 0.196 0.2 0.227 ...
$ atemp     : num  0.364 0.354 0.189 0.212 0.229 ...
$ hum       : num  0.806 0.696 0.437 0.59 0.437 ...
$ windspeed : num  0.16 0.249 0.248 0.16 0.187 ...
```

$ casual : int 331 131 120 108 82 88 148 68 54 41 ...

$ registered: int 654 670 1229 1454 1518 1518 1362 891 768 1280 ...

$ cnt : int 985 801 1349 1562 1600 1606 1510 959 822 1321 ...

1.2 DATA PARTITION

Two types of data partitions were done based on model analysis:

Type	Train Set	Test Set
1 (Used in many models)	Year 2011	Year 2012
2 (Used in one model)	Year 2011 – Mid 2012	Mid-Year 2012 – Year 2012 End

Please note that we used separate csv for different models as team members used different names and transformations in their respective csv's. Everyone used the copy of Daily.csv provided as part of assignment.

1.3 FEATURE ENGINEERING

Feature analysis was performed on various variables using Time series line chart and outlier analysis using the Box plots as shown below in Fig (1) and Fig (2)

Fig (1) Time Series Plot

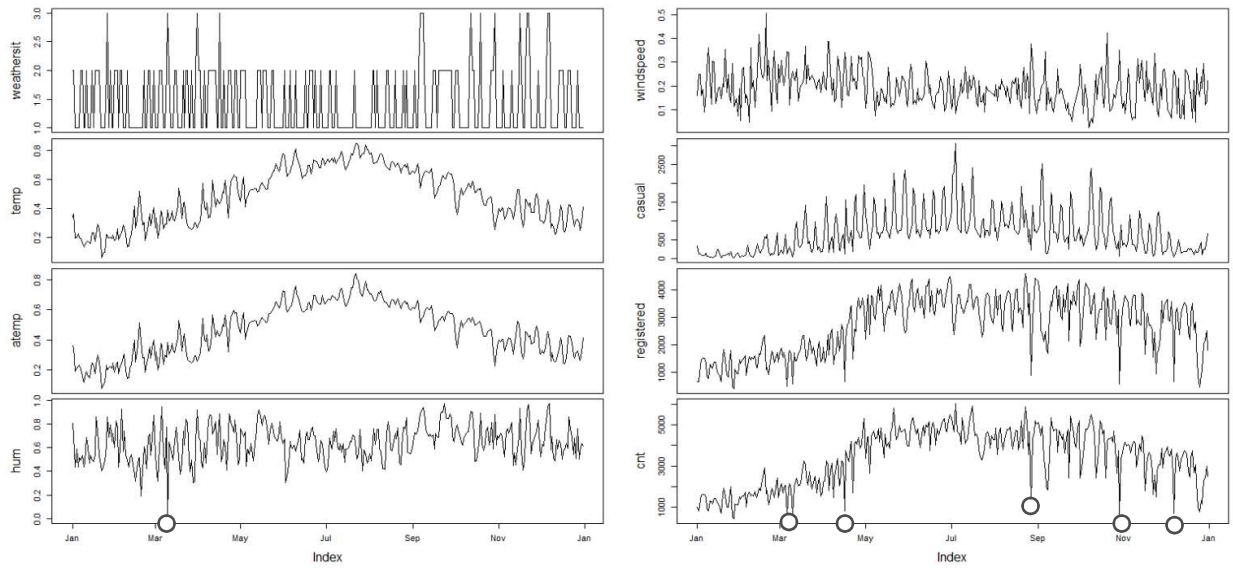


Fig (2) Box Plot Chart

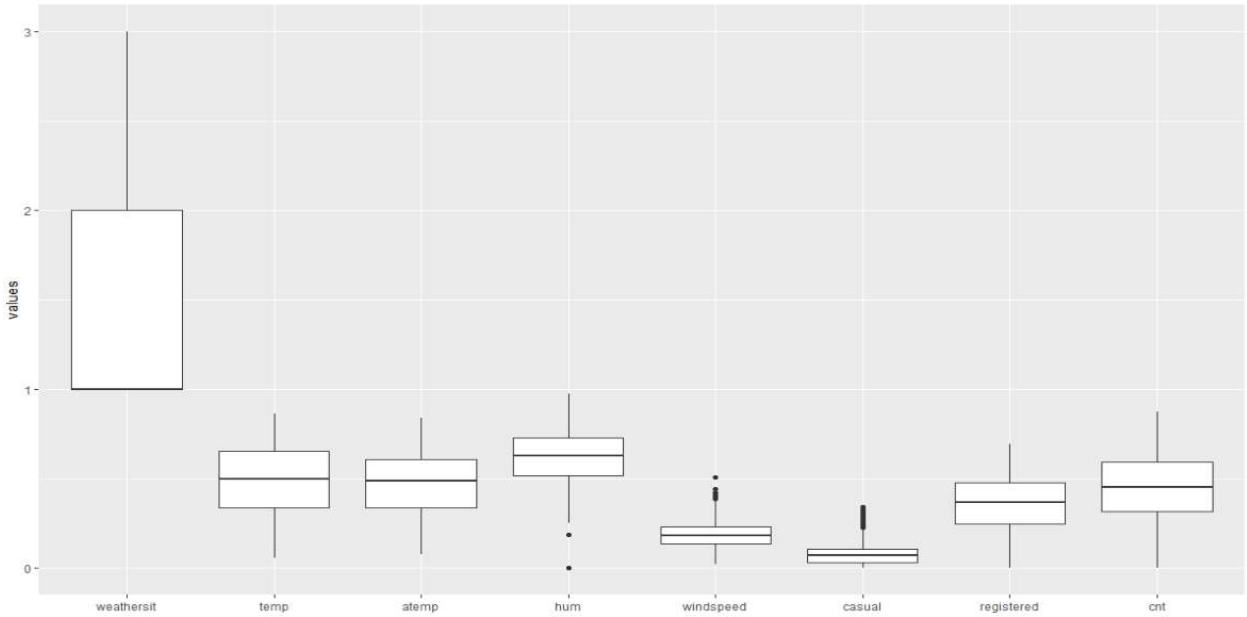
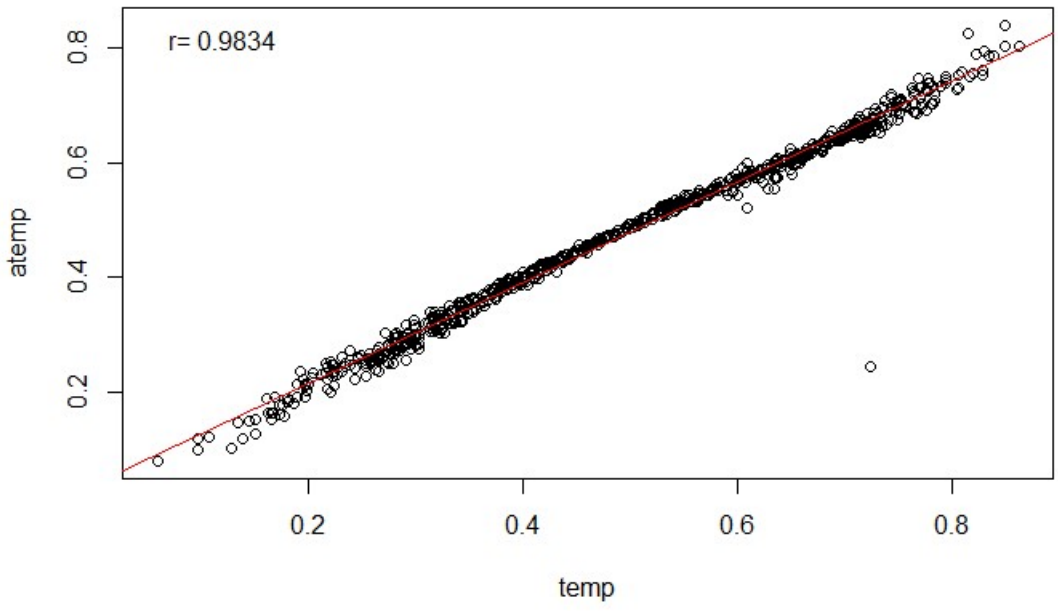


Fig (3) Correlation between temp and atemp



Here is the summary of our analysis and action taken:

- No variables found to have missing values
- humidity parameter has one zero value. We replaced this value with past 7 days moving average
- cnt, casual and registered variable have multiple unusual drops. Given the geographic location of Washington D.C. This could be due to harsh weather conditions. A simple google search showed up some instances e.g. in January. This data was imputed with 7 days moving average after trial with 1 to 10 days of moving average.
- We added 7 days moving average for weather parameters as well.
- temp and atemp are highly correlated variable (Fig 3). So we used atemp in our model
- We created a diff variable of temp and atemp by reversing the normalizing process. as *atemp*50 - temp*41*

1.4 MODEL BUILDING

While the business problem is of demand forecasting for maximizing the profit, it can be approached in multiple ways. We used different kinds of variables (Day, Weather and Demand) in different stages of modeling. This differential treatment also ensures that the predictions from different techniques are less correlated. This will help create a robust predictor as the Ensembling input.

Predictions are evaluated using total profit and RoI for business performance. Six different analytical techniques were used in this project. The common philosophy behind these techniques is that we break the prediction problem in two parts. We assume that there is an underlying demand pattern which does not depend on the Day or Weather variables. Given the nature of industry, it is expected (and observed) to be an increasing pattern with a strong seasonality and yearly effect. We forecast this using univariate time series analysis techniques on the demand variables.

There is a realization that the trend remains same for consecutive years but the demand for recent year is inflated as compared to previous one. Thus, training the demand variable using previous year and testing the same on following years must be accompanied by normalization to balance excessive volatility. Also, different models were created each for different types of demand (casual and registered) as we believe that casual and registered follow different patterns. Finally the output were added and normalized to predict the cnt.

1.4.1 Generalized Linear Models

As a benchmark, we started with **glm** which is used to fit generalized linear models, specified by giving a symbolic description of the linear predictor and a description of the error distribution. It was quite evident that there is a strong weekly pattern and thus weather related fields were replaced with last 7 days mean to get better prediction. The casual and final count demand was also considered as one of the predictor followed by normalization to balance excessive volatility in 2012.

Guided by Gaussian family, code snippet is below:

```
# Creating two output variables each for casual and registered in order to create
# different models for each of them. Creating single model on cnt will not lead to better prediction.
bikeshare_train[,output1 := casual]
bikeshare_train[,output2 := registered]

# Training the model for target variable casual using GLM
mylogit <- glm(output1 ~ season+workingday+weathersit+atemp+feel_diff+hum+windspeed+casual_lag, data = bikeshare_train, family = "gaussian")
bikeshare_test$output1 <- predict(mylogit, newdata = bikeshare_test, type = "response")

# Training the model for target variable registered using GLM
mylogit <- glm(output2 ~ season+workingday+weathersit+atemp+feel_diff+hum+windspeed+registered_lag, data = bikeshare_train, family = "gaussian")
bikeshare_test$output2 <- predict(mylogit, newdata = bikeshare_test, type = "response")
```

1.4.2 Random Forest

Random forests combine the predictions of multiple decision trees. The dataset is repeatedly divided into subtrees, guided by the best combination of variables. However, finding the right combination of variables can be difficult. For instance, a decision tree constructed based on a small sample might be not be generalizable to future, large samples. To overcome this, multiple decision trees could be constructed, by randomizing the combination and order of variables used. The aggregated result from these forests of trees would form an ensemble, known as a random forest. Weather related fields were replaced with last 7 days mean to get better prediction. The casual and final count demand was also considered as one of the predictor followed by out normalization to balance excessive volatility in 2012.

Code snippet is below:

```
# Creating two output variables each for casual and registered in order to create
# different models for each of them. Creating single model on cnt will not lead to better prediction.
bikeshare_train[,output1 := casual]
bikeshare_train[,output2 := registered]

# Training the model for target variable casual using RandomForest
random_forest <- randomForest(output1 ~season+workingday+weathersit+atemp+feel_diff+hum+windspeed+casual_lag, data = bikeshare_train)
bikeshare_test$output1 <- predict(random_forest, newdata = bikeshare_test, type = "response")

# Training the model for target variable registered using RandomForest
random_forest <- randomForest(output2 ~season+workingday+weathersit+atemp+feel_diff+hum+windspeed+registered_lag, data = bikeshare_train)
bikeshare_test$output2 <- predict(random_forest, newdata = bikeshare_test, type = "response")
```

1.4.3 Neural Network

Based on experience, Random forests may require more data but they almost always come up with a pretty robust model. On the other side, neural network algorithms require "relatively" large datasets to work well.

Code snippet is below:

```
# Creating two output variables each for casual and registered in order to create
# different models for each of them. Creating single model on cnt will not lead to better prediction.
bikeshare_train[,output1 := casual]
bikeshare_train[,output2 := registered]

# Training the model for target variable casual using Stacking
nn <- nnet(output1~season+workingday+weathersit+atemp+feel_diff+hum+windspeed+casual_lag,data=bikeshare_train, hidden=1, size = 10, decay=0.01, maxit=1000, linout=TRUE)
# Predicting the output for individual target casual
bikeshare_test$output1 <- predict(nn, bikeshare_test)

# Training the model for target variable registered using Stacking
nn <- nnet(output2~season+workingday+weathersit+atemp+feel_diff+hum+windspeed+registered_lag,data=bikeshare_train, hidden=1, size = 10, decay=0.01, maxit=1000, linout=TRUE)
# Predicting the output for individual target registered
bikeshare_test$output2 <- predict(nn, bikeshare_test)
```

1.4.4 Ensemble Stacking

It can take time to find well performing machine learning algorithms for dataset. This is because of the trial and error nature of applied machine learning. Once someone has a shortlist of accurate models, you can use algorithm tuning to get the most from each algorithm. Another approach that one can use to increase accuracy on dataset is to combine the predictions of multiple different models together. Stacking is a process of building multiple models (typically of differing types) and supervisor model that learns how to best combine the predictions of the primary models. GLM was used as a supervisor model that best combined the predictions from rpart, glm and knn.

Code snippet is below:

```
# Creating two output variables each for casual and registered in order to create
# different models for each of them. Creating single model on cnt will not lead to better prediction.
bikeshare_train[,output1 := casual]
bikeshare_train[,output2 := registered]

# Defining the Ensemble and algorithm list
myControl <- trainControl(method="repeatedcv", number=10, repeats=3, savePredictions=TRUE, classProbs=TRUE)
algorithmList <- c('rpart', 'glm', 'knn')
set.seed(123)

# Training the model for target variable casual using Stacking
models_casual <- caretList(output1~season+workingday+weathersit+atemp+feel_diff+hum+windspeed+casual_lag, data=bikeshare_train, trControl=myControl, methodList=algorithmList)

# Training the model for target variable registered using Stacking
models_reg <- caretList(output2~season+workingday+weathersit+atemp+feel_diff+hum+windspeed+registered_lag, data=bikeshare_train, trControl=myControl, methodList=algorithmList)

# Clubbing the different models using glm
stack.glm_casual <- caretStack(models_casual, method="glm", trControl=myControl)
stack.glm_reg <- caretStack(models_reg, method="glm", trControl=myControl)
print(stack.glm_casual)
print(stack.glm_reg)

# Predicting the output for individual targets, casual and registered
bikeshare_test$output1 <- predict(stack.glm_casual, newdata = bikeshare_test, type = "raw")
bikeshare_test$output2 <- predict(stack.glm_reg, newdata = bikeshare_test, type = "raw")
```

1.4.5 Decision Trees

Under the Individual category, we also made model using the Decision Tree. We built mainly 2 models:

1. Using weather and demand variables
2. Create the Moving average of 5 days and used 1.5 years as train data

Overall, weather variables were found to be actually used in the construction and business and model performance for the same is mentioned in the respective tables below.

```
```{R}
fit <- rpart(traindata_frame_na$target.cnt~traindata_frame_na$season + traindata_frame_na$holiday
             + traindata_frame_na$weekday + traindata_frame_na$workingday + traindata_frame_na$weathersit
             + traindata_frame_na$atemp + traindata_frame_na$hum + traindata_frame_na$windspeed+traindata_frame_na$casual,
   method="anova", data=traindata_frame_na)
printcp(fit)

# Function that returns Root Mean Squared Error
rmse <- function(error)
{
    sqrt(mean(error^2))
}

# Function that returns Mean Absolute Error
mae <- function(error)
{
    mean(abs(error))
}
```
```

1.4.6 Time Series Analysis – ARIMA (1,1,1)

We also performed the Univariate Time series analysis using ARIMA models. We took the total count (cnt) variable and prepared model by following the flow chart below:

Flow Chart to perform ARIMA

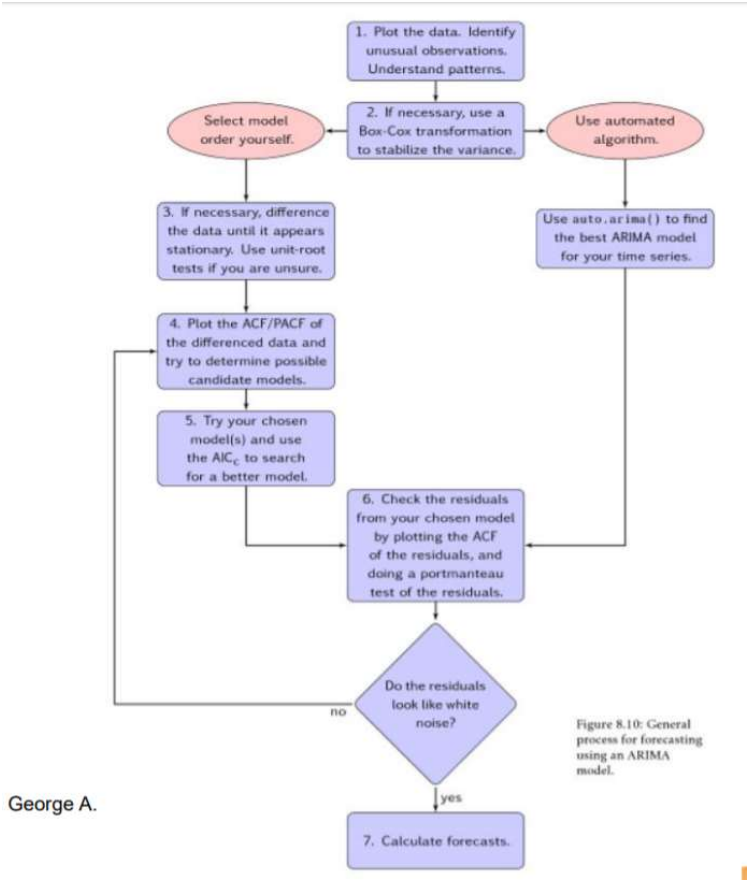


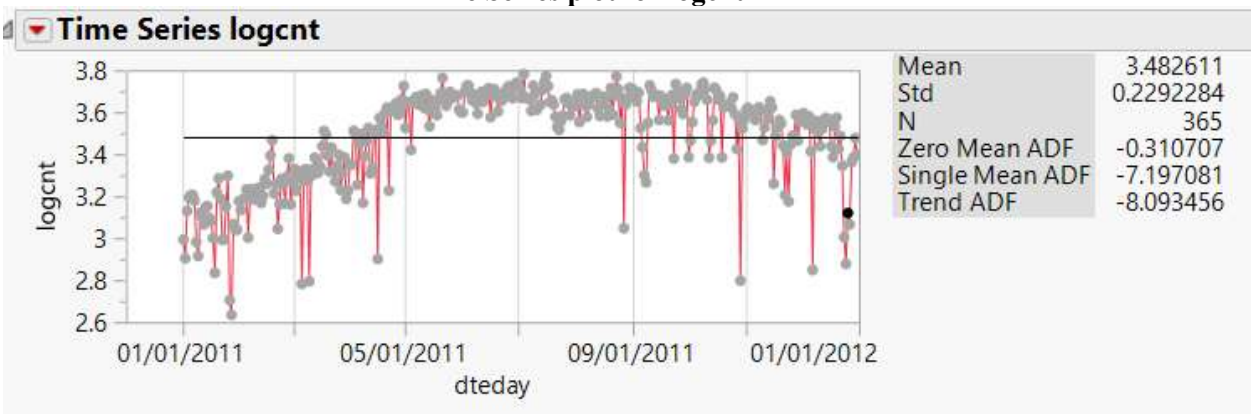
Figure 8.10: General process for forecasting using an ARIMA model.

Observations:

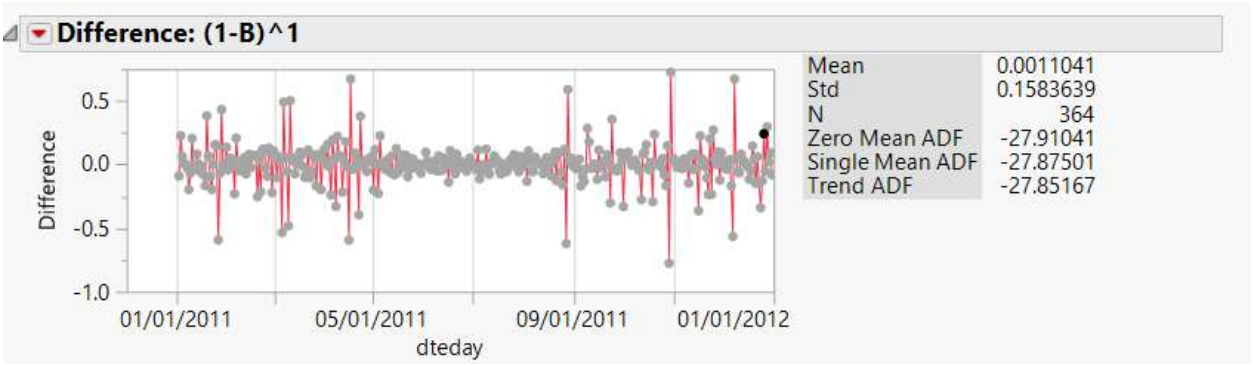
1. In first iteration, we used the cnt variable as it is and plotted the time-series
2. We used 1 order differencing and applied ***Dickey-Fuller Test***. Data output seems to be stationary
3. We then used the ACF and PACF plot to see the ARIMA order
4. Finally, we ran the ARIMA (1,1,1) model. Residual coefficient plot *(Ljung-Box Q test)* still showed significance at *p-value*
5. Hence, we decided to transform the cnt variable by taking log(cnt) to dampen the effect of seasonality and random effect.
6. We followed the step (1-4) again and found that this time we achieved the White Noise and a good AIC value to consider it as a model.
7. We used the model at Step 6 for the forecasting. Below is the Jmp output of our final model:

Team Ensemble: Kriti, Muni, Pooja, Pradeep, Sambit

Time Series plot for logcnt



Difference order 1



ACF and PACF Plot

Lag	AutoCorr	Ljung-Box Q	p-Value	Lag	Partial
0	1.0000	.	.	0	1.0000
1	-0.3652	48.9503	<.0001*	1	-0.3652
2	-0.1169	53.9818	<.0001*	2	-0.2888
3	0.0162	54.0791	<.0001*	3	-0.1816
4	0.0260	54.3298	<.0001*	4	-0.0984
5	-0.1095	58.7775	<.0001*	5	-0.1997
6	0.1025	62.6870	<.0001*	6	-0.0533
7	-0.0856	65.4205	<.0001*	7	-0.1541
8	0.0157	65.5122	<.0001*	8	-0.1169
9	-0.0022	65.5140	<.0001*	9	-0.1240
10	0.0608	66.9074	<.0001*	10	-0.0439
11	-0.0468	67.7332	<.0001*	11	-0.0658
12	0.0102	67.7722	<.0001*	12	-0.0666
13	0.0035	67.7769	<.0001*	13	-0.0458
14	-0.0248	68.0120	<.0001*	14	-0.0871
15	0.0466	68.8403	<.0001*	15	-0.0046
16	-0.0724	70.8485	<.0001*	16	-0.1168
17	0.0353	71.3268	<.0001*	17	-0.0558
18	0.0754	73.5181	<.0001*	18	0.0409
19	-0.0504	74.4991	<.0001*	19	-0.0012
20	0.0067	74.5162	<.0001*	20	0.0398
21	0.0234	74.7295	<.0001*	21	0.0452
22	-0.0733	76.8235	<.0001*	22	-0.0178
23	-0.0331	77.2508	<.0001*	23	-0.0876
24	0.0782	79.6462	<.0001*	24	-0.0125
25	-0.0842	82.4323	<.0001*	25	-0.1189

Team Ensemble: Kriti, Muni, Pooja, Pradeep, Sambit

ARIMA Model Summary

Model: ARIMA(1, 1, 1)

Model Summary

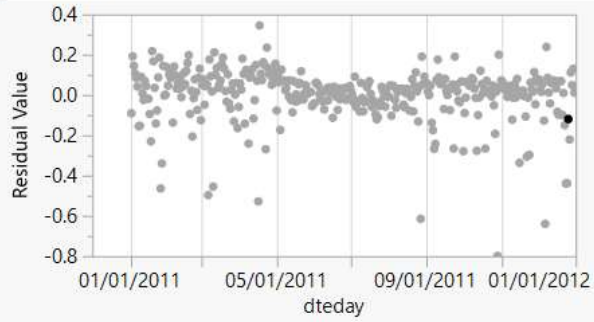
DF	361	Stable	Yes
Sum of Squared Errors	6.28720558	Invertible	Yes
Variance Estimate	0.01741608		
Standard Deviation	0.13197		
Akaike's 'A' Information Criterion	-437.1721		
Schwarz's Bayesian Criterion	-425.48064		
RSquare	0.6671109		
RSquare Adj	0.66526664		
MAPE	2.62026729		
MAE	0.08681359		
-2LogLikelihood	-443.1721		

Parameter Estimates

Term	Lag	Estimate	Std Error	t Ratio	Prob>|t|	Constant Estimate	Mu
AR1	1	0.24489110	0.0579974	4.22	<.0001*	0.00058561	0.00077553
MA1	1	0.89535828	0.0234048	38.26	<.0001*		
Intercept	0	0.00077553	0.0009710	0.80	0.4250		

Residual Plot

Residuals



Lag	AutoCorr	-.8 -.6 -.4 -.2 0 .2 .4 .6 .8	Ljung-Box Q	p-Value	Lag	Partial	-.8 -.6 -.4 -.2 0 .2 .4 .6 .8
0	1.0000		.	.	0	1.0000	
1	0.0027		0.0027	0.9584	1	0.0027	
2	-0.0176		0.1161	0.9436	2	-0.0176	
3	0.0250		0.3462	0.9511	3	0.0251	
4	0.0031		0.3498	0.9864	4	0.0027	
5	-0.0993		4.0102	0.5480	5	-0.0985	
6	0.0365		4.5047	0.6087	6	0.0370	
7	-0.0780		6.7759	0.4526	7	-0.0828	
8	-0.0016		6.7768	0.5609	8	0.0058	
9	0.0125		6.8356	0.6542	9	0.0084	
10	0.0601		8.1956	0.6097	10	0.0546	
11	-0.0160		8.2920	0.6869	11	-0.0089	
12	0.0149		8.3759	0.7551	12	-0.0001	
13	0.0118		8.4284	0.8146	13	0.0148	
14	-0.0023		8.4303	0.8657	14	-0.0069	
15	0.0465		9.2541	0.8639	15	0.0589	
16	-0.0210		9.4233	0.8950	16	-0.0288	
17	0.0655		11.0728	0.8528	17	0.0824	
18	0.0936		14.4455	0.6996	18	0.0901	
19	-0.0132		14.5129	0.7530	19	-0.0127	
20	-0.0004		14.5130	0.8036	20	0.0108	
21	-0.0088		14.5432	0.8452	21	-0.0238	
22	-0.0858		17.4138	0.7401	22	-0.0645	
23	-0.0459		18.2367	0.7445	23	-0.0411	
24	0.0560		19.4670	0.7266	24	0.0594	
25	-0.0155		19.5615	0.7694	25	-0.0050	

1.5 MODEL PERFORMANCE

Below table and plot shows the error plot for year 2012:

Model	Type	RMSE	MAE
Naive	Default		
Decision Trees	Default Variables	2549	2310
Decision Trees	Weather 5 days moving average+1.5 years train data	3575	3044
ARIMA	7 day Moving average	95.35284	70.26753
GLM		1275.46	934.6744
Random Forest	7 day Moving average	1616.523	1377.843
NNet	7 day Moving average	1877.859	1611.204
Ensemble (Stacking)	7 day Moving average	1237.905	934.270

Fig 4. Error plot (Test Data) for Decision Tree Prediction over time

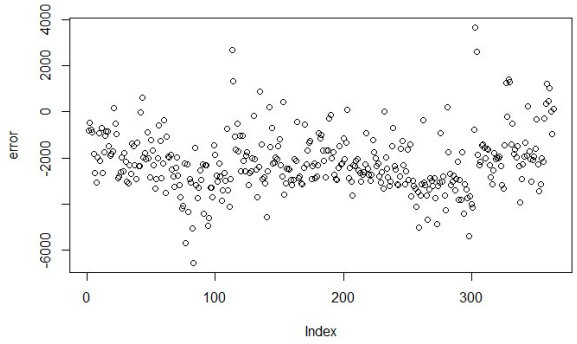


Fig 5 Error Plot GLM

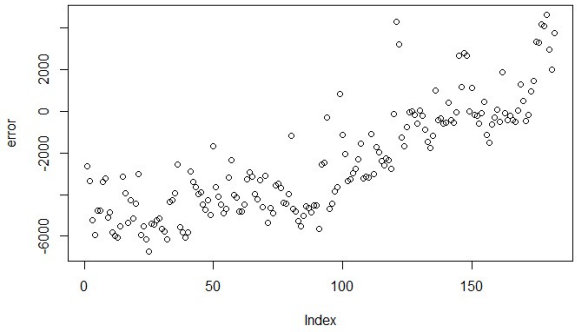
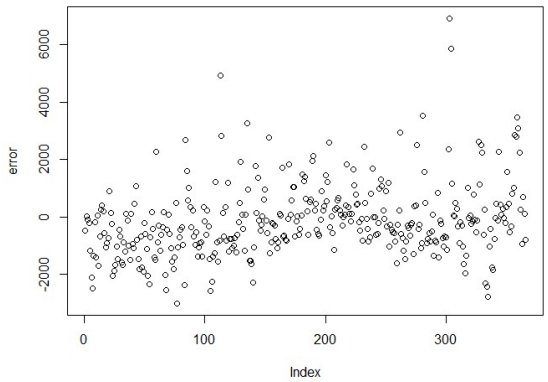


Fig 6 Error plot for Random Forest



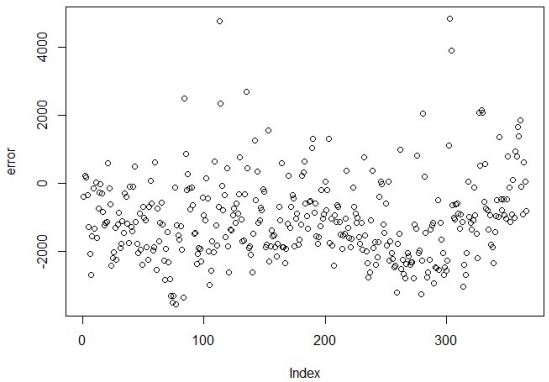


Fig 5 Error Plot NNet

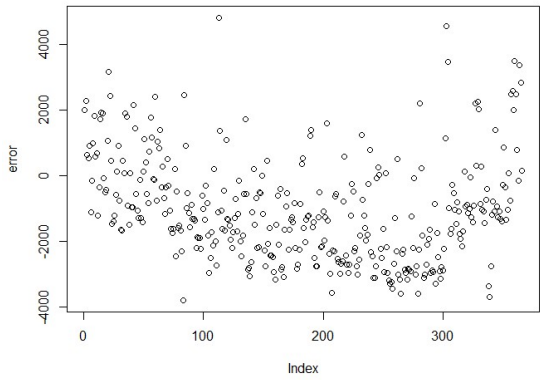
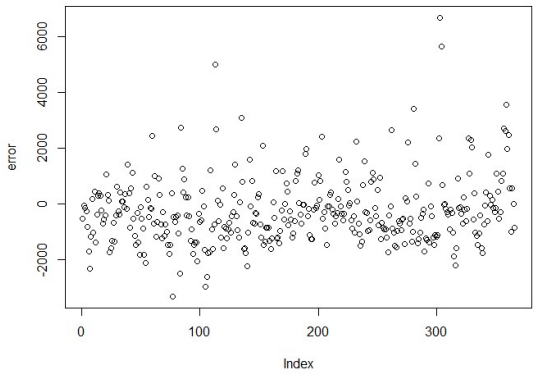


Fig 6 Error plot for Ensemble (Stacking)



1.6 BUSINESS PERFORMANCE

Below shows the return on investment comparisons of different models.

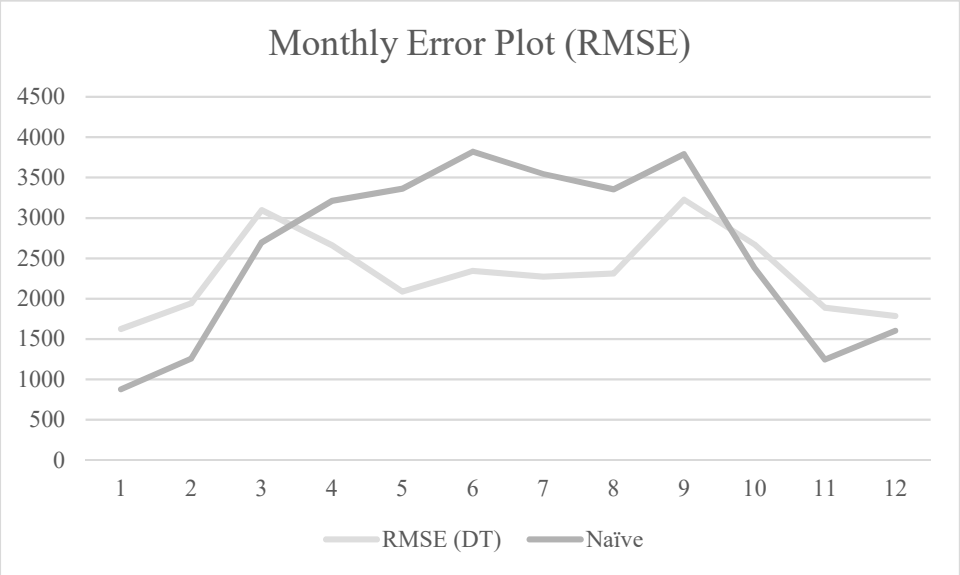
Table 2. Time Series Model Performance

Model	Type	Total Spend(,000)	Profit/Loss(,000)	ROI(%)
Naïve (Default)	Default	$4092	$1442	35.23
Ensemble (Stacking)	7 days moving averages for input variables Target variable: Registered & Casual GLM+KNN+Rpart	$3919	**$1582**	40.0
GLM	7 days moving averages for input variable Target variable: registered & Casual	$4007	$**1560**	38.93
Random Forest (Bagging)	7 days moving averages for input variables	$3315	$**1482**	44.70
NNet	7 days moving averages for input variables Target variable registered and casual	$3111	$1342	43.13
Decision Tree	Default Variables	$2485	$1187	47.76
Decision Tree	Weather 5 days moving	$1327	$498 (6 months)	37.5

	average+1.5 years train data			
ARIMA (1,1,1)	Univariate time series analysis over total count	$2340	$1031	44%

Business Insights

- The above Table gives the model profit for 2012 expressed as a $ total and profit expressed as a percentage of total expenditure (total costs for the year).
- First row of the above table gives the profit (total and percentage of expenditure) for the default prediction.
- Our best performing model, Ensemble (stacking) always generted the better predictions in terms of Profit and ROI. We compared it between 3-5 dollars. However, some of our models like decision trees shown above never generated profit better than default.
- We didn't observe any correlation between our model performance with respect to season or similar factor. However for odd extreme values, model tends to follow irregular pattern
- No. it didn't show any deviation for test dataset. Our RMSE plot shown below indicates random pattern. Our best performing Ensebmle stacking showed a constant error performance for year 2012



- For our best performing models, 1 year train model performed best in terms of total profit. Below are the numbers when trained for 1 year and 1.5 years:

Model Type	Training Period	Test Period	Profit (,000)
GLM	1 year	Jul2012-Dec2012	$847
GLM	1.5 years	Jul2012-Dec2012	$579
Ensemble	1 year	Jul2012-Dec2012	$862
Ensemble	1.5 years	Jul2012-Dec2012	$587

- We didn't perform the data balancing but simply by looking at the 18 months' data we can see that 2/3rd of the data shows uptrend (Jan-Dec) and 1/3rd of the data (Jul-Dec) is showing downtrend. Hence model may be biased towards uptrend. Thus, the overall profit might go down due to Over-Prediction. We could use give high weightage to Jul-Dec data. Alternatively, suggested approach would be to continuous build the data as mentioned in report at 4pm every day.

1.7 REFERENCES

Lecture Notes: *ARIMA/Decision Trees/GLM*