IE 48B Project Report

Predicting the Sign of the Imbalance in the Turkish Electricity Markets

Alp Serdaroğlu 2017402063
Mehmet Bahadır Erden 2016402000
Muhammed Burak Gür 2016403198

Introduction

The purpose of this project is to predict system imbalance information of electricity market in Turkey. First, we have made time-series analysis representation and develop prediction models such as linear regression, k-NN classifiers and combination of decision trees and linear regression on the sample data and then apply a model which has the best accuracy, to live data from 09 January 2022 until 22 January 2022.

System imbalance is important for electricity distribution networks in a way that if there exists excess amount of electricity, it will damage distribution system. Already storing electricity is costly and excess energy will damage to system, it is crucial to balance between producing and consuming electricity. However, foreseeing electricity consumption for everyday is difficult and prediction accuracy is low, hence it is hard to see balanced system direction in an hourly manner when we look at the EPDK site.

Before going prediction models, giving short background information about Turkey electricity market progress will be useful. Distributor and supplier need to know customer demand to position themselves Bilateral Contracts Market. If distributors can predict electricity demand, they can maximize profit and can arrange their energy facilities for fine tuning. Every producer or electricity market participant should place bid/ask orders for 24 hours of the next day until 12PM of that day. Predictions of companies are generally far from the actual demand/supply due to renewable energy sources, forecasting electricity supply/demand is becoming hard, the companies need to rearrange this imbalance via Intra-Day Market, i.e., they sell or buy imbalanced productions or consumptions at this market. If producers cannot solve their imbalance problems with the help of Intra-Day market, EPİAŞ penalizes them about imbalances amount with maximum of MCP and WAP for the corresponding hour multiplied by 1.03. That's why, producers should be aware of bidding, forecasting and intra-day market transactions. Our aim is to predict sign of system direction as positive, balanced or negative, not hourly electricity demand. In other words, we have tried to predict sign of difference between the total volume of down and up regulations. If total down volume is greater than up volume, the class is positive (energy surplus), it is negative otherwise. Since EPİAŞ can provide these up and down regulations in numerical, we need to transform these into classification type. Negative and positive sign meanings are provided above, and balanced sign are created by looking the difference between up and down regulations is between -50 and 50. After giving some background information, it is time to discuss approaches that were made. Before going into deep, brief information about representations, models and

approaches will be given here. The detailed information and codes will be provided explicitly for each one. Also note that the type of report did not mention in project instructions, HTML file type was chosen, and each approach was made on different notebooks and these notebooks were reachable from the main report via links.

In this project, we need to handle unexpected situations which increase or decrease electricity demand or supply such as sudden increase in temperatures, malfunctions on energy facilities, Iran gas crisis etc. If we can predict such unexpected situations, our model can provide more accurate predictions. To do that, we have thought on that, and we have come up with some ideas. Firstly, we have found breakdown data set when searching “EPDK Şeffaflık” website. Then we have made some data manipulations on them and added to model as feature. Secondly, weather prediction data can provide also information because Turkey has huge potentials in terms of renewable energy especially hydroelectric and wind, thus weather can affect adversely these sources. Lastly, Bitcoin or Ethereum data can be useful in a way that if there is significant amount increase in these coins, the amount of mining rigs can be increase, hence increases energy consumption drastically. However last one was remained idea, any application was not performed.

For representation side, SAX and PCA were used. In addition to that, outlier detection was performed for outliers for each day to search what happened at that day. Special days, national and religious days were handled. Breakdown data can also be used for such outlier detections. Weather dataset was used for seasonal information and added as features. Lag versions of dataset was also added to feature set because it can increase model accuracy. For model training side, k-NN classifiers with distance measures, linear regression and combination of linear regression and decision trees mentioned at Assist. Prof. Baydogan article were performed to train our model. For training set dates from 01/01/2019 to 31/11/2021 and for test set 2-week interval 01/12/2021 – 14/12/2021 were used. Two different approaches were performed for such models to see which one is better for classification or regression. For classification side we have predicted 3-class information (negative, positive, balanced classes) of system direction labels (calculations were given above), and we have predicted amount of difference between up and down regulations in regression side. For comparison of these models, some measures were used such as baseline methods (7 Lags, 168 Lags) for every hour, accuracy and root mean square error.

Context

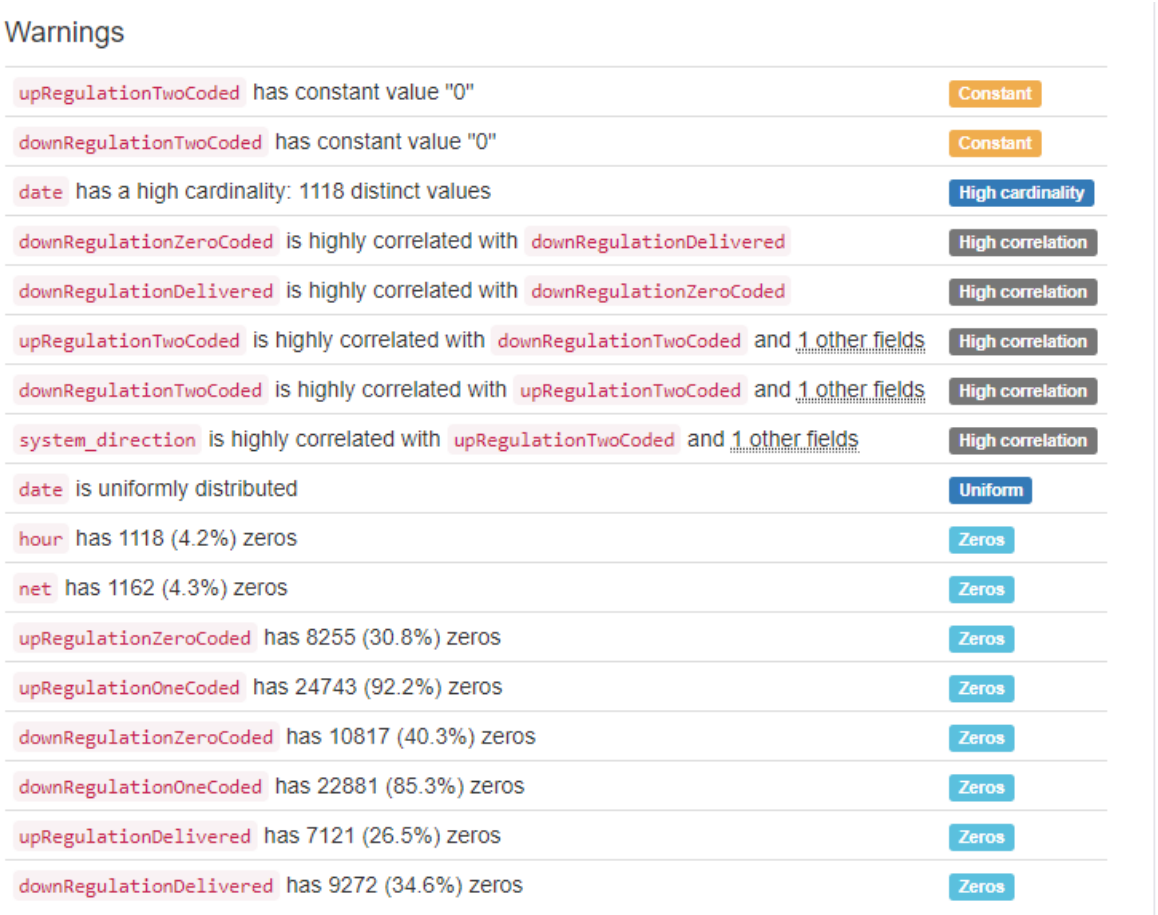
(In order to get specified techniques rapidly, a context part is added to Report.)

1. Data Analysis

In this section, 3 major phases related to data analysis exist, Descriptive analysis, anomaly detection, and hypothesis. The main aim of the analysis is to get useful information related to dataset to determine a path to get proper models for existing situation. Details of the phases will be given at the beginning of the sections.

A. Descriptive Analysis on Dataset

To get first impressions related to the first given data, pandas-profiling is used. In this library, warnings related to datasets were given. Warnings are as follows:

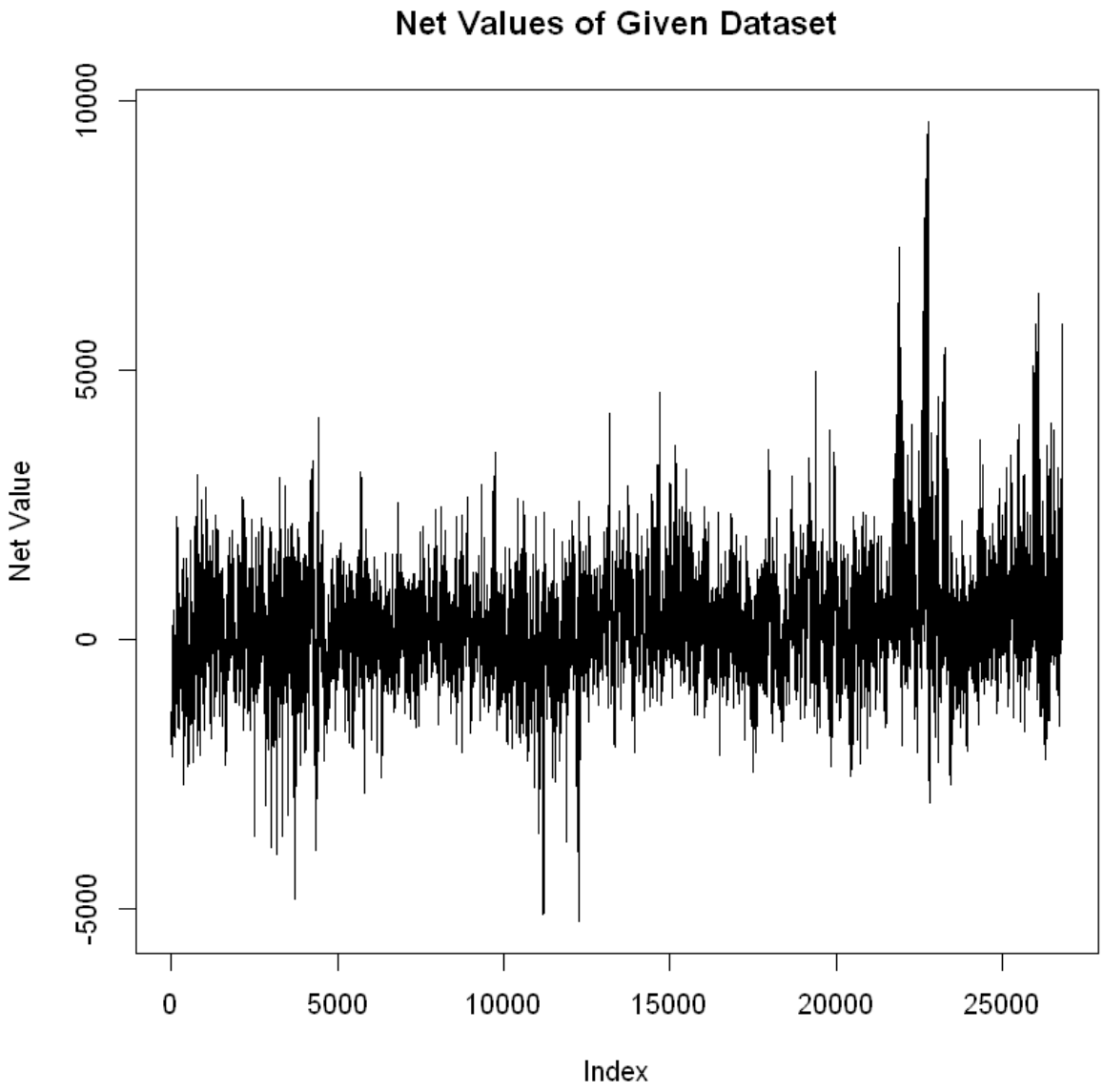


Important notes related to obtained warnings are:

- Both up and down regulation(two coded) has constant value 0.
- Up-regulation with one code has zero value with 92.2 percent.
- Down-regulation with one code has zero value with 85.3 percent.

So, it can be said that system direction can be analyzed by just looking at zero coded up and down regulations. In addition, important inspections related to the dataset is as follows:

In [14]:
```
plot(data$net,type="l", ylab="Net Value", main="Net Values of Given Dataset")
```



By looking at net values, it can be said that there is a random walk behavior, which means it is hard to predict the net value and find a relationship with target values. In addition, the constant variance assumption cannot be held for net value because there are some extreme values in different periods. Other visual inspections can be problematic by considering the number of instances for net value, so obtained statistical information related to net value must be inspected.

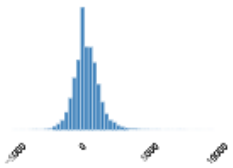
Statistical Information of Net Value

net
Real number (ℝ)

ZEROS

Distinct	25287
Distinct (%)	94.2%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%

Mean	291.7587979
Minimum	-5229.322
Maximum	9609.074
Zeros	1162
Zeros (%)	4.3%
Memory size	209.8 KiB



Toggle details

Statistics | Histogram | Common values | Extreme values

Quantile statistics

Minimum	-5229.322
5-th percentile	-1171.06675
Q1	-287.16825
median	184.95
Q3	799.17
95-th percentile	1936.7964
Maximum	9609.074
Range	14838.396
Interquartile range (IQR)	1086.33825

Descriptive statistics

Standard deviation	1056.485466
Coefficient of variation (CV)	3.621092058
Kurtosis	8.029920756
Mean	291.7587979
Median Absolute Deviation (MAD)	546.9875
Skewness	1.303666289
Sum	7828472.065
Variance	1116161.54
Monotocity	Not monotonic

First of all, histogram of the net value resembles a normal distribution. Therefore, it can be said that the linear regression assumption of normal distribution of target value is satisfied in this case. In addition, the mean of the net value is 291.758, so there is a positive tendency in the net value. 4.3 percent(1162 cases) of the dataset has zero net value. However, net value does not represents our final target, so we also need to control system direction variable.

Statistical Information of System Direction

Overview | Categories | Words | Characters

Value	Count	Frequency (%)
positive	14979	55.9%
negative	9408	35.1%
neutral	2421	9.0%

55.9 percent of the final target variable is positive and 35.1 percent is negative in the imbalance dataset. Namely, there is an imbalance problem in the target variable.

B. Anomaly Detection

Before getting into the details, anomaly detection was held to remove the effect of extreme points. This situation is not shown in the class, but the main aim is to apply learned models with a proper dataset. In addition, the removal of extreme points would not be applied by considering the mentioned fact. 2 different anomaly detection will be tried to understand features of the extreme days. The first method assumes the net value has a normal distribution and finds 2 points, lower and upper bounds, by using the quantile function. 0.05 is selected for our dataset to find enough extreme points.

In addition, an event list is created by looking at the formal holiday in Turkey, and obtained days are controlled with these days.

Extreme Days with Holidays

In [60]:
```
neg_ext_df[neg_ext_df["event"]!=""].iloc[:,[0,-1]]
```

Out[60]:

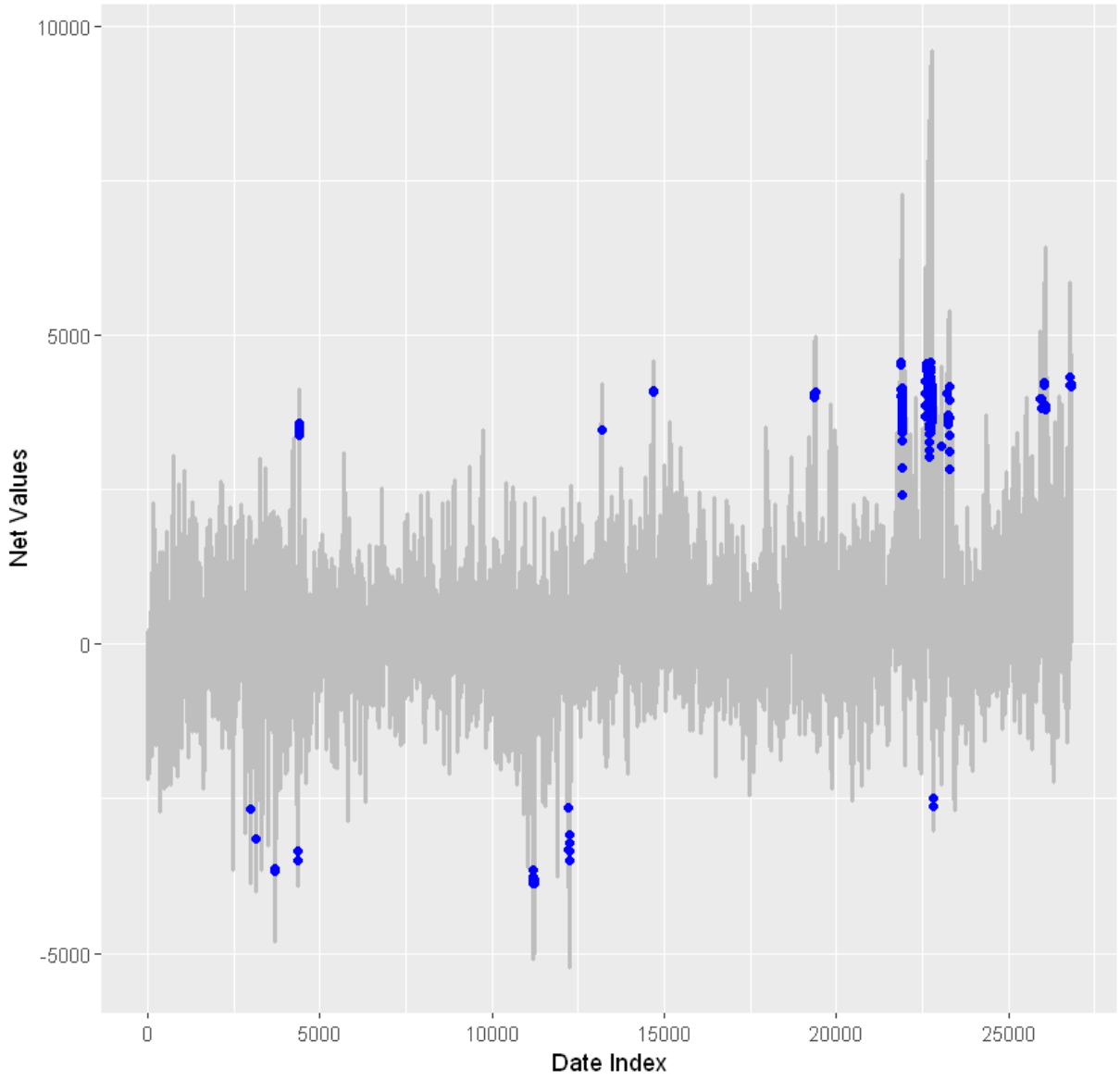
date	net	event
2019-06-04	-32237.438	Ramazan Bayrami
2019-06-05	-13244.768	Ramazan Bayrami
2020-05-01	-19906.628	Emek ve Dayanisma Gunu
2020-05-23	-20928.733	Ramazan Bayrami Arifesi
2020-05-24	-20373.771	Ramazan Bayrami
2020-05-25	-17973.323	Ramazan Bayrami
2020-08-03	-14282.613	Kurban Bayrami
2021-05-13	-16562.912	Ramazan Bayrami Arifesi

Formal Holidays exist only in the days with negative extreme values. In addition, there are limited holidays in the extreme days, and only some of the days of the sequential holidays represent

extreme values. Thus, it is hard to say that there is a certain effect of holidays. In addition, sequential extreme days are inspected and only Ramadan Holiday in 2020 is found important in this manner. The second method benefits from tsoutliers function in R to find outliers. The main problem in the second method is the decomposition of the net value includes some problems, which causes problems in the detection of outliers.

Anomaly Detection with tsoutliers Function

In [13]:
```
autoplot(ts(tsclean(net_val)), series="clean", color='red', lwd=0.7) +
  autolayer(ts(net_val), series="original", color='gray', lwd=1) +
  geom_point(data = tsoutliers(net_val) %>% as.data.frame(),
             aes(x=index, y=replacements), col='blue') +
  labs(x = "Date Index", y = "Net Values")
```



Obtained outlier points weren't related to formal holidays, so it is hard to relate the extreme points with effect of holidays. Details of the extreme points for both methods can be found in the noteboks

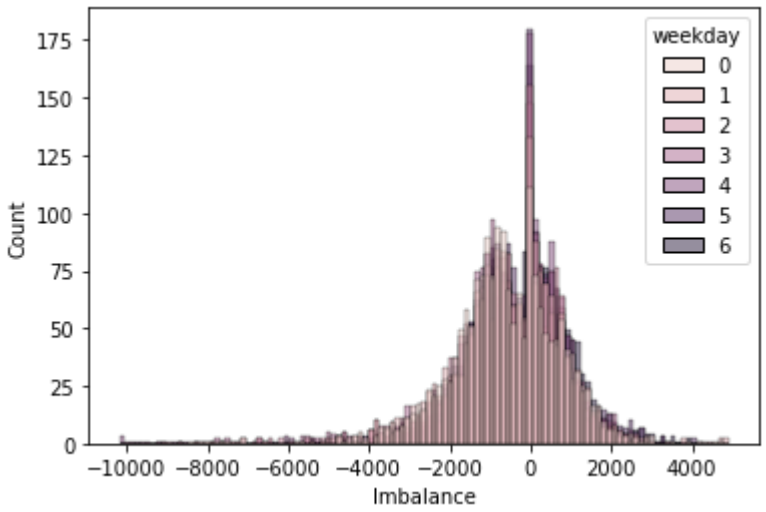
Data Control in R and Python.

C. Claims on Data

Before creating models and input datasets, different claims will be controlled with visual inspections and some basic statistics to determine model types and feature sets.

First Claim: Day is important in imbalance value

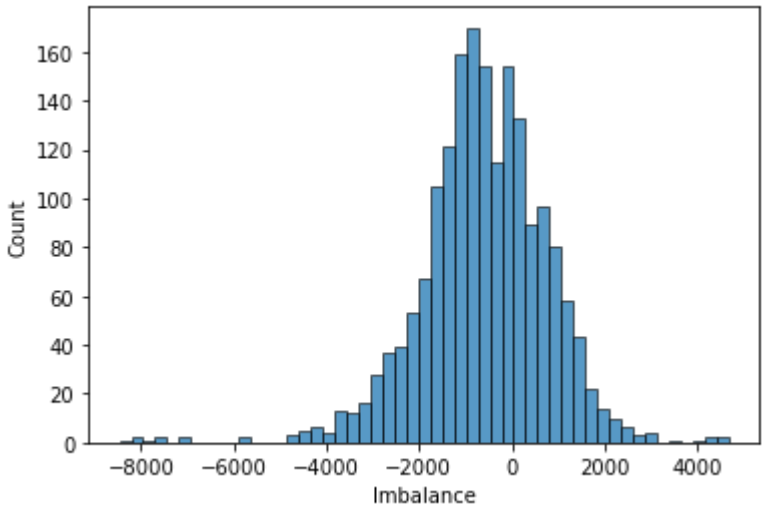
In [12]:
```
sns.histplot(data, x="Imbalance", hue="weekday")
plt.show()
```



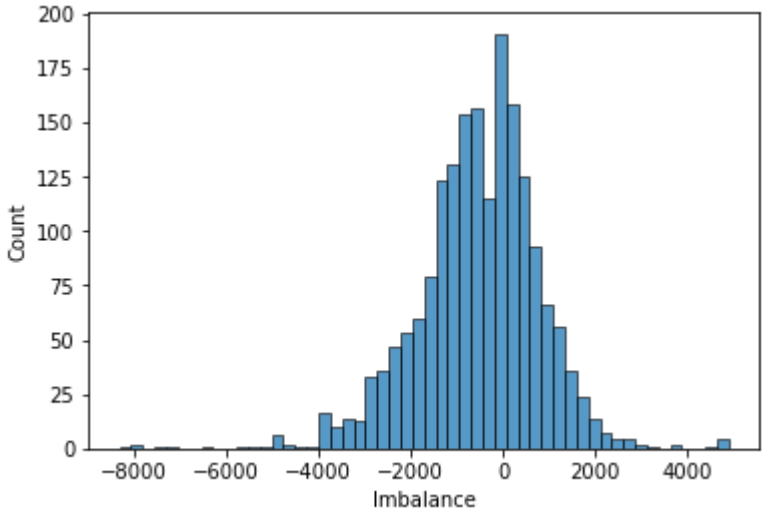
By looking at the histogram of the Imbalance values of the days, there is no accurate change in the behavior of imbalance value. An example for how to detailed inspection in this manner can be found as follows:

In [187]:
```
for i in range(7):
    print("Day:", i)
    hist_sns(data, i)
```

```
Day: 0
Mean Imbalance at day 0: -612.6164106753813
Std Imbalance at day 0: 1372.7417829217982
```

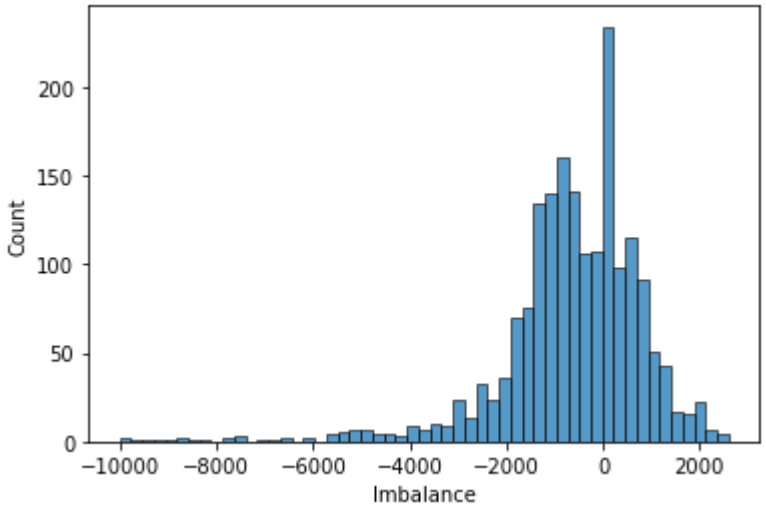



```
Day: 1
Mean Imbalance at day 1: -556.4544588744589
Std Imbalance at day 1: 1350.962138043434
```

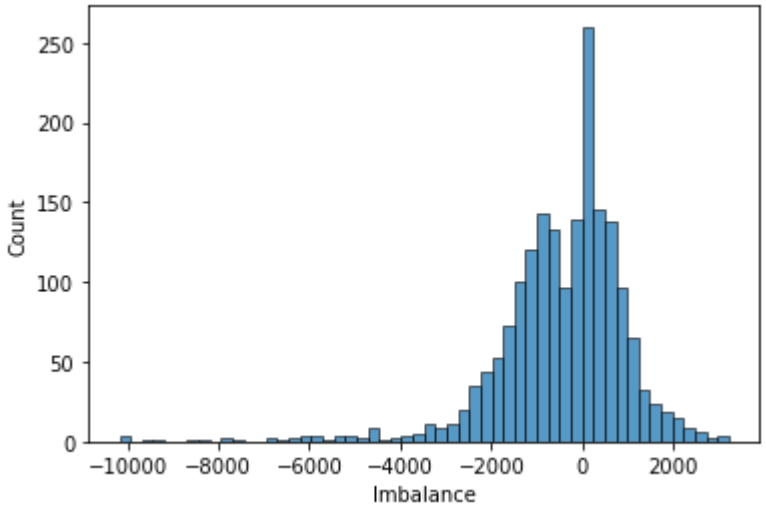



```
Day: 2
Mean Imbalance at day 2: -622.9021536796537
Std Imbalance at day 2: 1439.0517592314645
```


```
Day: 3
Mean Imbalance at day 3: -646.0348160173161
Std Imbalance at day 3: 1492.1527422893305
```

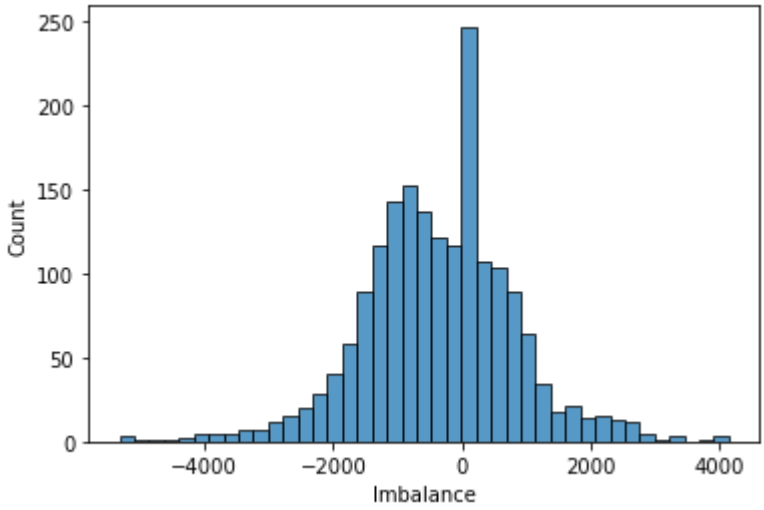



```
Day: 4
Mean Imbalance at day 4: -463.7532738095238
Std Imbalance at day 4: 1425.9323499182244
```

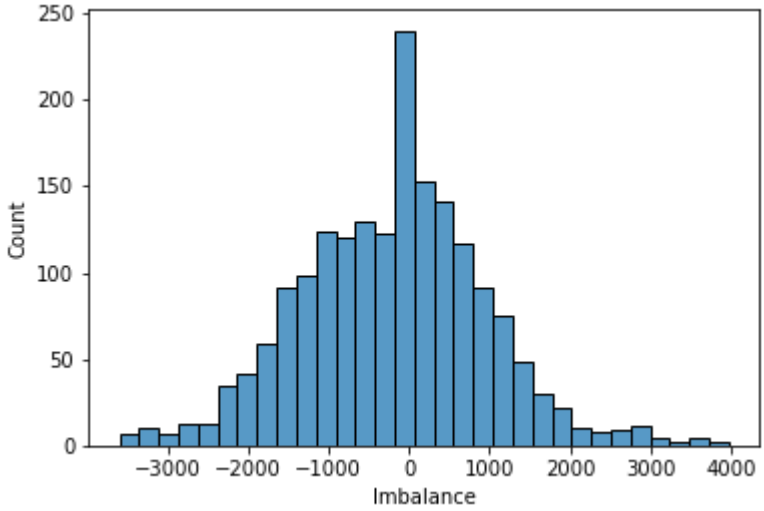



```
Day: 5
```

```
Mean Imbalance at day 5: -366.5577559912854
Std Imbalance at day 5: 1180.2793476663537
```




```
Day: 6
Mean Imbalance at day 6: -208.16169389978214
Std Imbalance at day 6: 1146.6680466709397
```

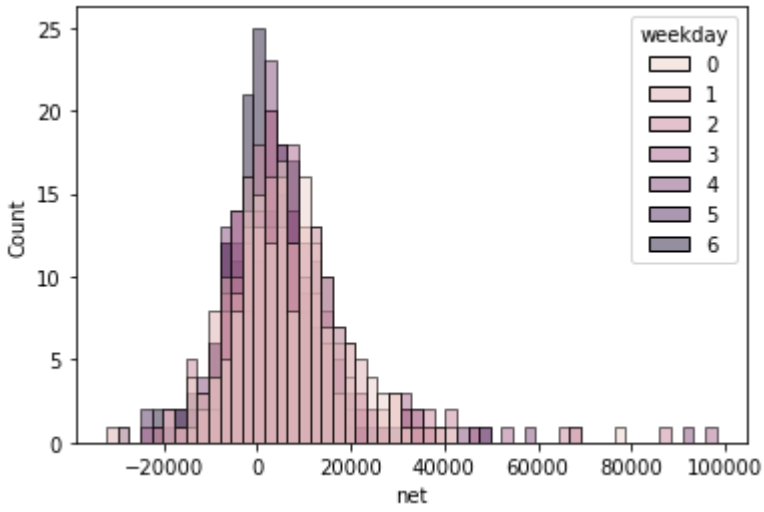


According to statistical information and plots demonstrate that Monday, Tuesday, Wednesday, Thursday are similar, but Friday, Saturday, and Sunday have different behaviors. This type of detailed plots will be applied for coming claims, but they will not be added to the report to make it

shorter. These mentioned plots can be found in the notebook Claims in Python.

Second Claim: Day is important in imbalance value in daily level

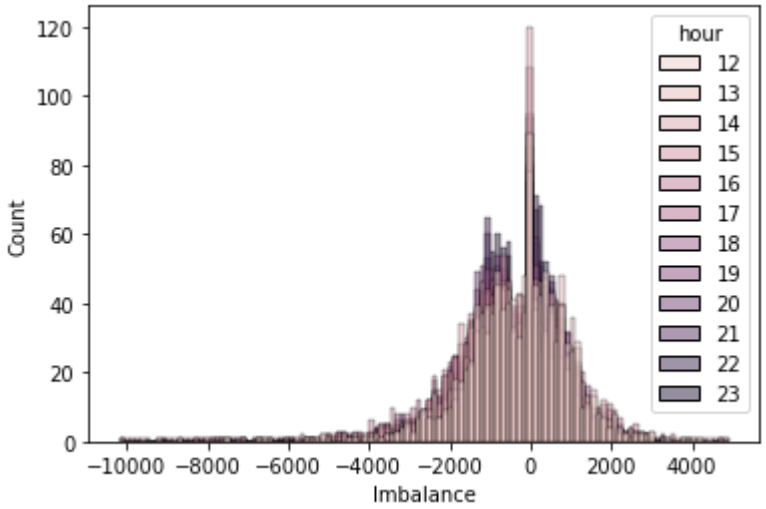
In [50]:
```
sns.histplot(daily_data, x="net", hue="weekday")
plt.show()
```



Again, a similar result is achieved in this perspective. Daily prediction approach includes additional process to find hourly prediction, so this claim will be dismissed.

Third Claim: Hour is important in imbalance value

In [55]:
```
sns.histplot(data, x="Imbalance", hue="hour")
plt.show()
```



This plot is one of the most important plots to determine the type of models in the prediction because if there is a change with hour information, multiple models can be conducted to get better results in the predictions. However, it is hard to say that there is a certain difference between the

hours. Namely, a general model can be a better option to have more instances to train a model and generalize the model for all cases.

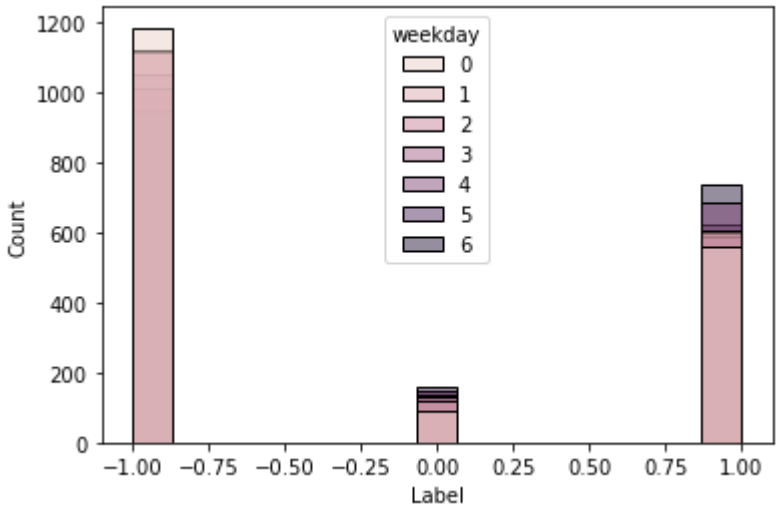
Detailed plots as in the daily level also suggest this situation, but there are still some hours distinguishing from others. These hours will be marked in the main models to give this distinction information, but this will not cause to the creation of distinct models to predict these hours. Obtained results are as follows:

- 17, 18, 19, and 20 resemble each other
- 12, 13, 22, and 23 have different behaviors compared to other hours

This information allows us to decrease the number of columns in the models' input data and better generalization in the input. In addition, it will alleviate problems of dummy encoding.

Fourth Claim: Day is important in Actual Target(System Direction)

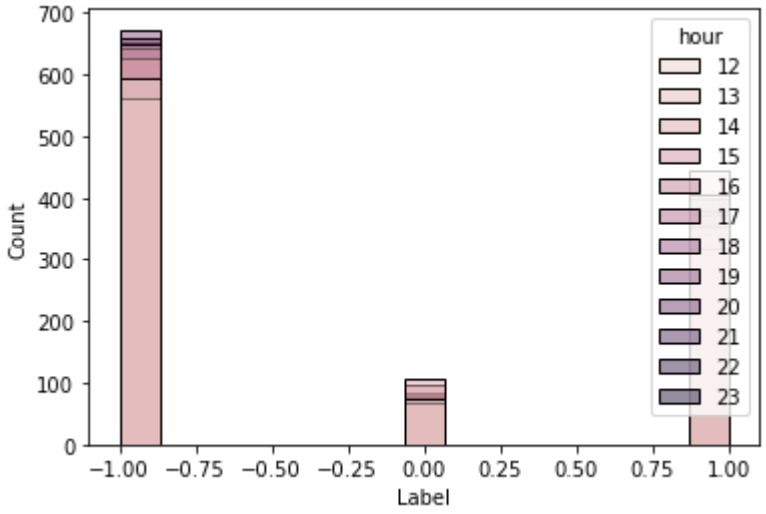
In [58]:
```
sns.histplot(data, x="Label", hue="weekday")
plt.show()
```



Similar results with Imbalance values are obtained in this visual inspection. Much as the behavior of the days resembles each other, there are still some differences for some days. Obtained these differences are coherent with determined claims in the first inspection.

Fifth Claim: Hour is important in Actual Target(System Direction)

In [62]:
```
sns.histplot(data, x="Label", hue="hour")
plt.show()
```



Similar results with Imbalance values are obtained in this visual inspection. Much as the behavior of the days resembles each other, there are still some differences for some days. Obtained these differences are coherent with determined claims in the third inspection.

Lastly, the transition in the system direction was controlled by creating a table, similar to the Markov chain. By creating this table, transition processes would have been controlled. However, obtained results have very problematic performance. So, this inspection will not be added to the report, but this attempt can be found at the claims in Python Notebook.

2. Feature Preparation

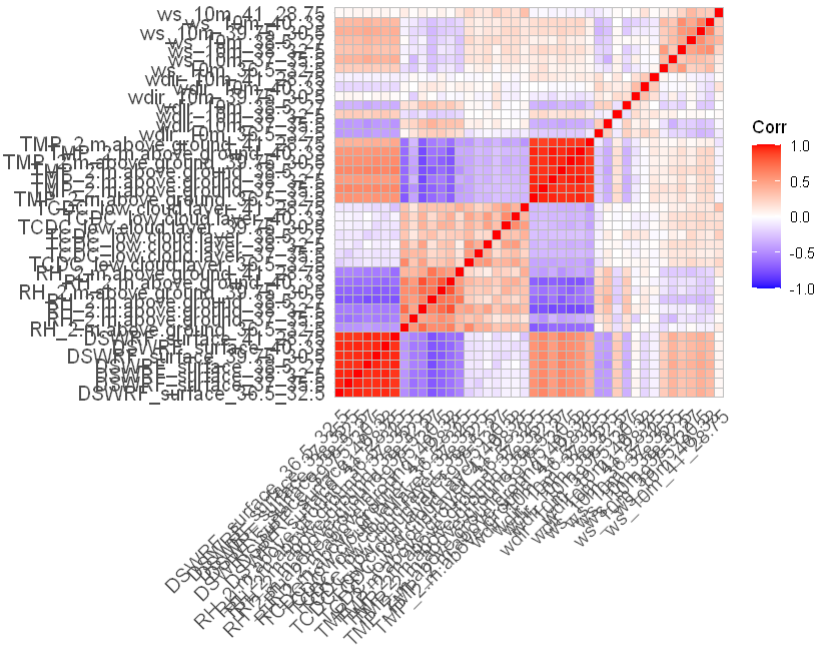
A. Weather Dataset

In addition to the original feature set, weather data is used in some of the models. The weather data set includes several meteorlogical metrics measured in six different locations. The data set covers a time period starting from the 31 December 2018. During our period of analysis, this data set is updated at regular intervals. It is assumed that it includes actual readings for the past and

forecasts for the future periods. During the training of our models it is assumed that the actual meteorological readings do not differ significantly from the actual measurements thus, when the actual reading is not available forecasted values are used to make predictions.

Initial analysis of the variables showed that the values from the same group are correlated highly with each other. Since this can cause multicollinearity problems in our models, PCA is used to reduce their dimensionality and eliminate the multicollinearity problem.

In [5]:
```
corr <- cor(wide_feat[,-c('date', 'hour')])
ggcorrplot(corr)
```



PCA for the Weather Data

PCA is used for each of the measurement groups. The results for the first group which is the DSWRF (downward shortwave radiation at the surface) measurements from 7 different locations are displayed below.

Since a single component explains the 95% of the variance, only one component is included in the data set to replace seven columns.

In [13]:
```
pca_rep_dswrf = princomp(train_feat[,c(3:9)])
summary(pca_rep_dswrf) # one column covers 95%. One component added to the datase

train_feat[,DSWRF:=pca_rep_dswrf$scores[,1]]
test_feat[, DSWRF := predict(pca_rep_dswrf,test_feat)[,1]]
```

```
Importance of components:
                           Comp.1      Comp.2       Comp.3       Comp.4
Standard deviation     708.6134971 92.99214403 71.646787130 57.006486049
Proportion of Variance   0.9530437  0.01641294  0.009742881  0.006167974
Cumulative Proportion    0.9530437  0.96945662  0.979199506  0.985367480
                           Comp.5       Comp.6       Comp.7
Standard deviation     52.557724388 51.285446450 48.134915131
Proportion of Variance  0.005242846  0.004992088  0.004397586
Cumulative Proportion   0.990610325  0.995602414  1.000000000
```

Results of the PCA for RH (relative humidity) measurements are displayed below. Three components that account for the 87% of the variance are included in the data set.

In [15]:
```
pca_rep_rh = princomp(train_feat[,c(10:16)])
summary(pca_rep_rh) #three components cover 87% of the variance. Three components

train_feat[,RH1 := pca_rep_rh$scores[,1]]
test_feat[, RH1 := predict(pca_rep_rh,test_feat)[,1]]

train_feat[,RH2 := pca_rep_rh$scores[,2]]
test_feat[, RH2 := predict(pca_rep_rh,test_feat)[,2]]

train_feat[,RH3 := pca_rep_rh$scores[,3]]
test_feat[, RH3 := predict(pca_rep_rh,test_feat)[,3]]
```

```
Importance of components:
                          Comp.1     Comp.2      Comp.3      Comp.4     Comp.5
Standard deviation     44.3277571 17.3407962 15.88967907 12.98951215 9.40515854
Proportion of Variance  0.6793138  0.1039578  0.08728692  0.05833169 0.03058096
Cumulative Proportion   0.6793138  0.7832716  0.87055848  0.92889017 0.95947114
                          Comp.6     Comp.7
Standard deviation     7.80745378 7.50169915
Proportion of Variance 0.02107355 0.01945531
Cumulative Proportion  0.98054469 1.00000000
```

Results of the PCA for TCDC (total cloud cover at the low cloud layer) measurements are displayed below. Three components that account for the 74% of the variance are included in the data set.

In [16]:
```
pca_rep_tcdc = princomp(train_feat[,c(17:23)])
summary(pca_rep_tcdc) #three components cover 74% of the variance. Three componer

train_feat[,tcdc1 := pca_rep_tcdc$scores[,1]]
test_feat[, tcdc1 := predict(pca_rep_tcdc,test_feat)[,1]]

train_feat[,tcdc2 := pca_rep_tcdc$scores[,2]]
test_feat[, tcdc2 := predict(pca_rep_tcdc,test_feat)[,2]]

train_feat[,tcdc3 := pca_rep_tcdc$scores[,3]]
test_feat[, tcdc3 := predict(pca_rep_tcdc,test_feat)[,3]]
```

```
Importance of components:
                          Comp.1     Comp.2     Comp.3     Comp.4      Comp.5
Standard deviation     55.254647 31.9107340 26.5401198 21.7461332 21.16066793
Proportion of Variance  0.474474  0.1582517  0.1094663  0.0734918  0.06958787
Cumulative Proportion   0.474474  0.6327257  0.7421920  0.8156838  0.88527169
                            Comp.6      Comp.7
Standard deviation     20.20179583 18.16932657
Proportion of Variance  0.06342416  0.05130415
Cumulative Proportion   0.94869585  1.00000000
```

Results of the PCA for TMP (temperature at 2m above ground) measurements are displayed below. One component that account for the 95% of the variance is included in the data set.

In [18]:
```
pca_rep_tmp = princomp(train_feat[,c(24:30)])
summary(pca_rep_tmp) #one component cover 95% of the variance. One component adde

train_feat[,tmp := pca_rep_tmp$scores[,1]]
test_feat[, tmp := predict(pca_rep_tmp,test_feat)[,1]]
```

```
Importance of components:
                           Comp.1     Comp.2     Comp.3      Comp.4      Comp.5
Standard deviation     22.1118887 2.96218649 2.50093149 2.138838885 1.687046909
Proportion of Variance  0.9505693 0.01705913 0.01216006 0.008893818 0.005533328
Cumulative Proportion   0.9505693 0.96762846 0.97978852 0.988682335 0.994215663
                            Comp.6     Comp.7
Standard deviation     1.374200724 1.04250113
Proportion of Variance 0.003671406 0.00211293
Cumulative Proportion  0.997887070 1.00000000
```

Results of the PCA for WDIR (wind direction in degrees at 10 meters height) measurements are displayed below. No components are included .

In [23]:
```
pca_rep_wdir = princomp(train_feat[,c(31:37)])
summary(pca_rep_wdir) # results are not great. no components are added.

#train[,wdir := pca_rep_wdir$scores[,1]]
#test_feat[, wdir := predict(pca_rep_wdir,test_feat)[,1]]
```

```
Importance of components:
                           Comp.1      Comp.2     Comp.3     Comp.4     Comp.5
Standard deviation     129.1873147 111.1436270 91.8539955 83.7733605 80.9635305
Proportion of Variance   0.2717849   0.2011660  0.1373984  0.1142872  0.1067492
Cumulative Proportion    0.2717849   0.4729509  0.6103493  0.7246365  0.8313856
                           Comp.6      Comp.7
Standard deviation     75.1586465 68.59445129
Proportion of Variance  0.0919906  0.07662377
Cumulative Proportion   0.9233762  1.00000000
```

Results of the PCA for TMP (temperature at 2m above ground) measurements are displayed below. Three components that account for the 75% of the variance is included in the data set.

In [25]:
```
pca_rep_wda = princomp(train_feat[,c(38:44)])
summary(pca_rep_wda) #three components cover 74% of the variance. Three component

train_feat[,wda1 := pca_rep_wda$scores[,1]]
test_feat[, wda1 := predict(pca_rep_wda,test_feat)[,1]]

train_feat[,wda2 := pca_rep_wda$scores[,2]]
test_feat[, wda2 := predict(pca_rep_wda,test_feat)[,2]]

train_feat[,wda3 := pca_rep_wda$scores[,3]]
test_feat[, wda3 := predict(pca_rep_wda,test_feat)[,3]]
```

```
Importance of components:
                          Comp.1    Comp.2     Comp.3    Comp.4    Comp.5
Standard deviation     3.7617941 2.2267192 1.70395211 1.6890977 1.4577241
Proportion of Variance 0.4794615 0.1679943 0.09837359 0.0966659 0.0719970
Cumulative Proportion  0.4794615 0.6474559 0.74582944 0.8424953 0.9144923
                           Comp.6     Comp.7
Standard deviation     1.14773217 1.09837679
Proportion of Variance 0.04463184 0.04087581
Cumulative Proportion  0.95912419 1.00000000
```

B. Breakdown Dataset

To predict deviation from balance of energy production and consumption, we have found a dataset which consists of unexpected malfunctions of energy facilities when we were looking at "EPDK Şeffaflık" website. We have thought that these unexpected malfunctions can cause decreasing in productions and thereby system_direction can deviate from predictions. Such deviations and the working capacity of the failed energy facilities can give us an idea about how much deviation can

occur in that hour. To understand this, we applied a correlation matrix with our original dataset and checked whether there was a correlation. Unfortunately, we could not see a sufficient level of correlation to increase the accuracy of our model.

In [113]:
```
head(dat)
```

A data.table: 6 × 8

Organizasyon Adý	Santral Ýsmi	UEVÇB	Olay Baþlangýç Tarihi	Olay Bitiþ Tarihi	Ýþletmedeki Kurulu Güç	Olay Sýrasýnda Kapasite	Ge
<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	
ÇELÝKLER ORHANELÝ TUNÇBÝLEK ELEKTRÝK ÜRETÝM A.Þ.	ORHANELÝ tes	ORHANELÝ 154	28/12/2021 00:57	28/12/2021 23:59	210,00	0,00	KA B PAT
ABK ENERJÝ ELEKTRÝK ÜRETÝM A.Þ.	Söke-Çatalbük Rüzgar Elektrik Santrali	SÖKE ÇATALBÜK	26/06/2021 00:00	26/06/2021 23:59	30,00	28,00	T ar
DOÐAL ENERJÝ HÝZMETLERÝ SAN. VE TÝC. A.Þ.	DOÐAL ENR.BÝYOKÜTLE ENR.SANT.	DOÐAL ENERJÝ BES	06/03/2020 00:17	06/03/2020 00:59	5,20	0,00	bes
ELEKTRÝK ÜRETÝM A.Þ.	ELBÝSTAN-A TS	AFÞÝN A 1	27/07/2019 00:00	27/07/2019 23:59	340,00	0,00	ÜN T
BAKIR ENERJÝ ELEKTRÝK ÜRETÝM A.Þ.	KÜREKDAÐI RES	KÜREKDAÐI RES	08/12/2019 14:27	08/12/2019 15:25	32,50	0,00	t ar
YENÝ ELEKTRÝK ÜRETÝM A.Þ.	Yeni Doðalgaz Kombine Çevrim Santrali	Yeni Elektrik Doðalgaz Kombine Çevrim Santrali	09/01/2022 00:00	09/01/2022 23:59	865,00	0,00	KONT SÝST ARI

With data manipulation, nominal power lost at each hour and each day are calculated. Resulting data frame is displayed below.

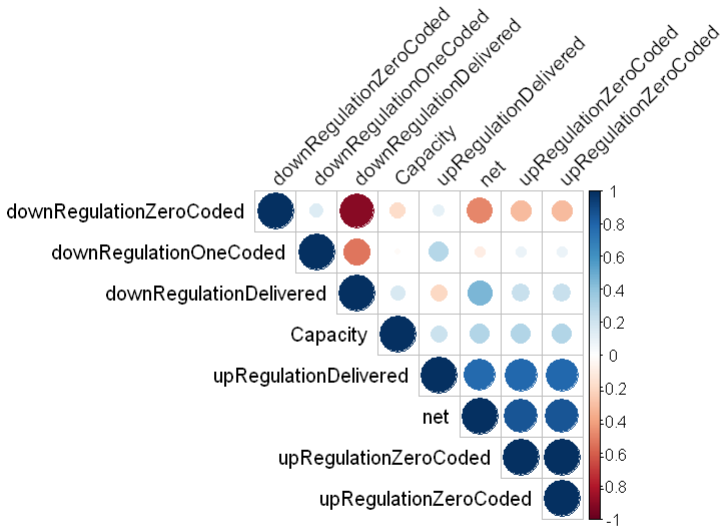
In [119]:
```
head(breakdown,10)
```

A data.table: 10 × 5

ID	Date	Hour	NominalPower	Capacity
<int>	<date>	<dbl>	<dbl>	<dbl>
1	2019-01-01	0	32	0
2	2019-01-01	0	155	0
2	2019-01-01	1	155	0
2	2019-01-01	2	155	0
2	2019-01-01	3	155	0
2	2019-01-01	4	155	0
2	2019-01-01	5	155	0
2	2019-01-01	6	155	0
2	2019-01-01	7	155	0
2	2019-01-01	8	155	0

Correlation matrix between "net" and "Capacity"

In [144]:
```
corrplot(correlation2, type = "upper", order = "hclust",
         tl.col = "black", tl.srt = 45)
```



C. Challenges

During this project, there were some challenges which limited our analysis capabilities and performance of our models. First, the data set that we aim to classify was unbalanced meaning there were considerably more positive observations compared to the negative observations. This caused our models to be biased and estimate most of the instances as positive resulting in low precision values for our models. This problem could have been countered with oversampling and undersampling methods such as SMOTE which will result in a balanced data set.

Overview | Categories | Words | Characters

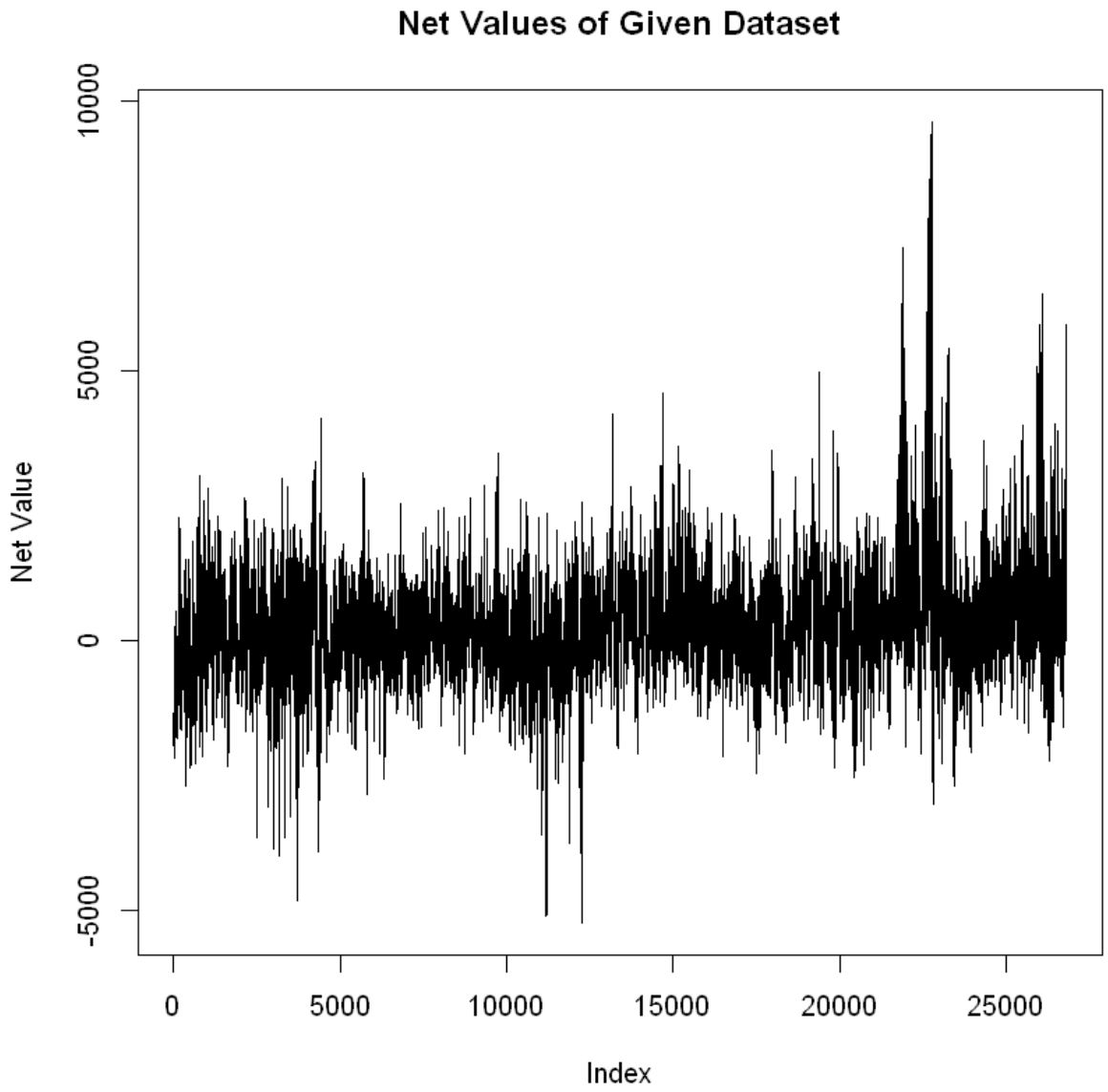
Value	Count	Frequency (%)
positive	14979	55.9%
negative	9408	35.1%
neutral	2421	9.0%

In addition, ranges of the labels were not similar to each other meaning the range of values for "Negative" and "Positive" labels compared to "Neutral" labels were considerably different than

each other. Range for the "Neutral" values were considerable small compared to other labels. This caused challenges in predicting a time interval as "Neutral" since there were very few number of instances which was classified as "Neutral" in the data set.

Moreover, the target variable, "net", looked significantly similar to a white noise series making it hard to predict or explain its behavior since there were no dominant trend or seasonal behavior. Furthermore, the lag while making our predictions affected the accuracy of our models significantly. While estimating 12:00, we had access to data at 05:00 at the time of submission which means 6 periods of data are missing from the analysis. The lag increases further while predicting other time periods. For example, for the hour of 23:00, lag increases to 17 periods which considerably affect the accuracy of our predictions. Finally, another challenge that we faced during our analysis were the lack of computational resources. Calculating a distance matrix for approximately 26 thousand instances substantial amount of time and computational effort which made it challenging for us to try different parameter values in a quick fashion. Also, it is not very easy to retrain the model every day in limited time before generating our predictions.

```
In [14]: plot(data$net,type="l", ylab="Net Value", main="Net Values of Given Dataset")
```



D. Assumptions

In the project, different assumptions were held to continue with a generalized approach. First of all, by looking at claims, the number of dummy variables has been decreased. In this case, grouped features that have similar attributes are assumed. Secondly, there is no absolute trend in the net value, so removing trend in data is not applied and a constant level is assumed. Similarly, it is hard to observe a decreasing or an increasing variance in the net value. Therefore, stable variance in the net value is assumed.

Moreover, there is no certain seasonal and cyclic feature in net value. So, no seasonal and cyclic components exist in the net value is assumed. In other words, the net value that resembles white noise is claimed with these assumptions, which can be also validated by looking plot of net value.

In addition, the given weather dataset includes a lot of correlated features, describing similar things. In this manner, PCA is applied. In general, variation in similar features was satisfied at least with 60 percent cumulative proportion with creating new principal components. How many new principal components will be created is determined by looking existing model and input dataset.

Lastly, only learned techniques/methods in the class have been conducted in the project. Nevertheless, fundamental techniques were used to continue the project. For example, PACF was used to determine the best lag value for the input dataset.

Approaches

For prediction models, we limited our approaches to what we saw in class. For these models, we used the weather forecasts and feature selection with PCA as the data set. We also wrote a function with many metric units of measurement in it that we use for model comparisons. In addition to the function we wrote, we checked confusion matrix of models both test data and train data set. Note that since there is unbalanced class information in dataset (positive imbalance is much more than negative imbalance), precisions of classes in prediction models are low, i.e., models can predict more accurate positive imbalances. First of all, we used different versions of the k-NN distance approximations such as k-NN with SAX representations and k-NN with different sizes, which are the most widely used in the course. Considering that the file size increases exponentially with the increase size of our dataset while making distance calculations, we passed the 2.4gb limit very easily. Also, we chose not to use dynamic time warping (DTW), LCSS and ERP instead of Euclidean distance since it would be computationally too long. Besides, we used decision trees which is widely used prediction algorithm. We also used 2 different approaches for these decision trees (classification and regression). Besides these decision trees, which are very logical in nature, we did not forget to include the method in Assoc. Prof. Baydogan's article named "Explainable boosted linear regression for time series forecasting". We aimed to increase the accuracy of our model by adding new features to our dataset, which we introduced into our linear regression model by using the decision trees method introduced by this model.

Several baseline methods are used for comparison purposes. First baseline method uses previous day's same hour's system direction which achieved an accuracy of 55.33%. Resulting confusion matrix is as follows:

In [96]:
```
d <- table(data$Label,data$b1_label)
d
sum(diag(d))/sum(d)
sum(d)
```

```
     -1    0    1
-1 9382 1129 3853
0  1105  284  796
1  3858  772 4597
```

0.553344196151459

25776

Second baseline method uses previous week's same hour's system direction which achieved an accuracy of 49.27%. Resulting confusion matrix is as follows:

In [97]:
```
d <- table(data$Label,data$b2_label)
d
sum(diag(d))/sum(d)
```

```
     -1    0    1
  -1 8603 1121 4605
  0  1089  253  836
  1  4552  800 3773
```

0.49270443196005

Decision Tree

First approach used to tackle this classification problem, is the use of decision trees. In this analysis, the feature set which was mentioned in the previous sections and the weather data set are used. Number of columns in the weather data set is reduced with the use of PCA.

Within this approach two different methods are considered. In the first method, a combination of feature and weather data set is used to construct the decision trees. Then, these decision trees are used to predict the sign of the net imbalabce. On the other hand, the second approach utilized the combination of decision trees and k-NN algorithm to build the model. Decision trees is used to extract features using the combination of original and weather data set. Then, this feature is combined with the windowed time series data set created for each hour. Finally, k-NN algorithm is used as a classifier.

First Method

The first method includes the direct use of decision trees to make predictions about the sign of the net imbalance. Features used in this analysis consists of the data set created in the previous sections along with the weather data set with PCA dimensionality reduction. A summary of the features used in this method are as follows:

In [99]:
```
head(training)
```

A data.table: 6 × 31

system_direction	lag_24	lag_168	is_monday	is_wday	is_friday	is_saturday	is_sunday	is_over	i
<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<
Positive	1776.793	-855.563	0	1	0	0	0	0	
Positive	1824.072	-78.651	0	1	0	0	0	0	
Positive	1948.549	202.696	0	1	0	0	0	0	
Positive	2024.501	206.927	0	1	0	0	0	0	
Positive	2276.499	-564.851	0	1	0	0	0	0	
Positive	1628.765	-1067.564	0	1	0	0	0	1	

Target variable of the model is the sign of the imbalance and all the variables in the feature set are included in the analysis. During the analysis, complexity parameter for the decision trees are set to 0 while different maximum depth parameters are used to determine the optimum model. To determine the best maximum depth parameter, 10-fold cross validation with 3 repetitions is used. Rpart2 model in the caret package is used for building models and parameter tuning with cross validation. Results of the model training for decision trees are as follows:

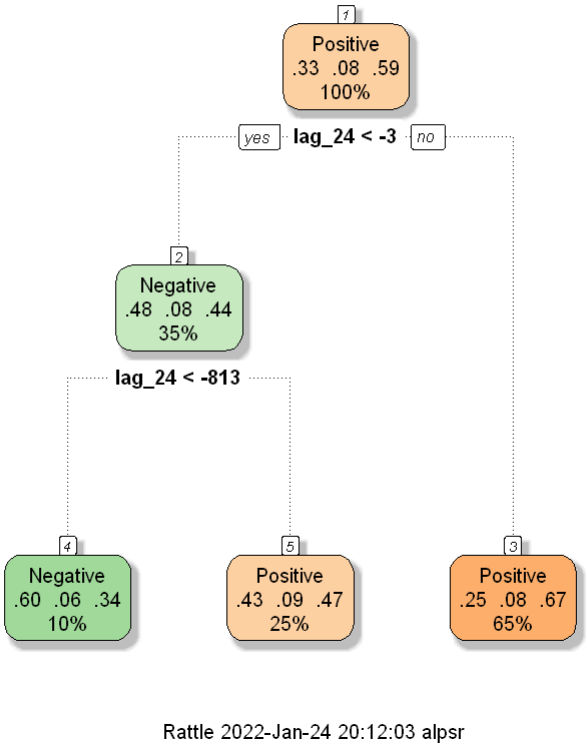
In [101]:
```
model1
```

```
CART

12696 samples
   30 predictor
    3 classes: 'Negative', 'Neutral', 'Positive'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 11426, 11427, 11427, 11427, 11427, 11427, ...
Resampling results across tuning parameters:

  maxdepth  Accuracy   Kappa
  1         0.6050208  0.1870097
  2         0.6116108  0.1184808
  6         0.6110588  0.1454270
  7         0.6110588  0.1454270
  8         0.6110588  0.1454270

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was maxdepth = 2.
```

The best maximum depth parameter is 2 which results in 61% accuracy over the validation set.

Then, the final models is trained over the whole training period using the a maximum depth of 6. Its summary can be found below.

In [121]:
```
dec_tree1 = rpart(system_direction ~ .,training,method='class',control=rpart.cont
fancyRpartPlot(dec_tree1)
```



Its performance over the training set is as follows:

In [125]:
```
confusionMatrix(data = as.factor(train_pred_label), reference = as.factor(trainc
```

```
Confusion Matrix and Statistics

          Reference
Prediction Negative Neutral Positive
  Negative      771      79      434
  Neutral         0       0        0
  Positive     3468     925     7019

Overall Statistics

               Accuracy : 0.6136
                 95% CI : (0.605, 0.6221)
    No Information Rate : 0.587
    P-Value [Acc > NIR] : 5.797e-10

                  Kappa : 0.1189

 Mcnemar's Test P-Value : < 2.2e-16

Statistics by Class:

                     Class: Negative Class: Neutral Class: Positive
Precision                    0.60047             NA          0.6151
Recall                       0.18188        0.00000          0.9418
F1                           0.27920             NA          0.7441
Prevalence                   0.33388        0.07908          0.5870
Detection Rate               0.06073        0.00000          0.5529
Detection Prevalence         0.10113        0.00000          0.8989
Balanced Accuracy            0.56061        0.50000          0.5519
```

Finally, its performance over the test is as follows:

In [126]:
```
confusionMatrix(data = as.factor(test_pred_label), reference = as.factor(testclas
```

```
Confusion Matrix and Statistics

          Reference
Prediction Negative Neutral Positive
  Negative       21       6       55
  Neutral         2       0        2
  Positive       14       5       63

Overall Statistics

               Accuracy : 0.5
                 95% CI : (0.422, 0.578)
    No Information Rate : 0.7143
    P-Value [Acc > NIR] : 1

                  Kappa : 0.078

 Mcnemar's Test P-Value : 4.305e-06

Statistics by Class:

                     Class: Negative Class: Neutral Class: Positive
Precision                     0.2561        0.00000          0.7683
Recall                        0.5676        0.00000          0.5250
F1                            0.3529            NaN          0.6238
Prevalence                    0.2202        0.06548          0.7143
Detection Rate                0.1250        0.00000          0.3750
Detection Prevalence          0.4881        0.02381          0.4881
Balanced Accuracy             0.5510        0.48726          0.5646
```

The model achieved a relatively better performance over the training set compared to other models. However, its performance decreased considerably over the test set which suggests that there is overfitting in the model

Second Method

The second method includes the combination of decision trees and k-NN approach to build models. First, decision trees used in a similary way to the first method. However, the decision trees are used to construct a feature for the k-NN algorithm. After constructing the decision trees, the node information for each data and hour is added as a feature to the windowed time series data set.

The windowed time series data set includes 8 observations from the previous day for each hour. For example for hour 12:00, observations from 11:00 of the previous day to 18:00 of the previous day are included in the data set. Similarly, for hour 13:00, observations from 12:00 of the previous day to 19:00 of the previous day are included in the data set. Finally, for hour 23:00 observarions from 22:00 of the previous day to 05:00 of the same day are included in the data set. Final feature set used in the analysis is displayed below:

In [113]:
```
head(training_window)
head(testing_window)
```

A data.table: 6 × 12

date	hour	t1	t2	t3	t4	t5	t6	t7	t8	syst
<date>	<int>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	
2019-01-08	12	1928.721	1776.793	1824.072	1948.549	2024.501	2276.499	1628.765	1427.524	
2019-01-08	13	1776.793	1824.072	1948.549	2024.501	2276.499	1628.765	1427.524	1173.955	
2019-01-08	14	1824.072	1948.549	2024.501	2276.499	1628.765	1427.524	1173.955	145.134	
2019-01-08	15	1948.549	2024.501	2276.499	1628.765	1427.524	1173.955	145.134	-35.417	
2019-01-08	16	2024.501	2276.499	1628.765	1427.524	1173.955	145.134	-35.417	-392.690	
2019-01-08	17	2276.499	1628.765	1427.524	1173.955	145.134	-35.417	-392.690	92.241	

A data.table: 6 × 12

date	hour	t1	t2	t3	t4	t5	t6	t7	t8	sy
<date>	<int>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	
2021-12-01	12	-76.756	-311.200	-768.558	-945.563	-1530.633	-839.498	-432.022	-500.934	
2021-12-01	13	-311.200	-768.558	-945.563	-1530.633	-839.498	-432.022	-500.934	-265.800	
2021-12-01	14	-768.558	-945.563	-1530.633	-839.498	-432.022	-500.934	-265.800	-260.893	
2021-12-01	15	-945.563	-1530.633	-839.498	-432.022	-500.934	-265.800	-260.893	-466.796	
2021-12-01	16	-1530.633	-839.498	-432.022	-500.934	-265.800	-260.893	-466.796	-458.977	
2021-12-01	17	-839.498	-432.022	-500.934	-265.800	-260.893	-466.796	-458.977	699.439	

The decision tree, with a maximum depth of 8, constructed on the feature set is as follows. Based on this decision tree each hour and day in the dataset is assigned a node id which is then included in the windowed time series data set. Then this, windows time series data set is fed into k-NN classifier. To determine the optimum value for k, 10-fold cross-validation with 3 repetitions is used. Optimum value for k found to be 13 with a training accuracy of 59.8%. Results of the k-NN classifier is as follows:

In [115]:
```
model2
```

```
k-Nearest Neighbors

12696 samples
    9 predictor
    3 classes: 'Negative', 'Neutral', 'Positive'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 11427, 11426, 11427, 11427, 11427, 11425, ...
Resampling results across tuning parameters:

  k   Accuracy   Kappa
   5  0.5668457  0.1230520
   7  0.5807873  0.1305218
   9  0.5842532  0.1258439
  11  0.5902918  0.1309286
  13  0.5937573  0.1327221

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 13.
```

Then, the best model is tested over the test period resulting in the following confusion matrix. Testing accuracy found to be 70.24%.

In [124]:
```
confusionMatrix(data = as.factor(testing_window$system_direction), reference = as
```

```
Confusion Matrix and Statistics

          Reference
Prediction Negative Neutral Positive
  Negative       10       0       26
  Neutral         1       0        7
  Positive       16       0      108

Overall Statistics

               Accuracy : 0.7024
                 95% CI : (0.6271, 0.7704)
    No Information Rate : 0.8393
    P-Value [Acc > NIR] : 1.00000

                  Kappa : 0.14

 Mcnemar's Test P-Value : 0.01559

Statistics by Class:

                     Class: Negative Class: Neutral Class: Positive
Precision                    0.27778        0.00000          0.8710
Recall                       0.37037             NA          0.7660
F1                           0.31746             NA          0.8151
Prevalence                   0.16071        0.00000          0.8393
Detection Rate               0.05952        0.00000          0.6429
Detection Prevalence         0.21429        0.04762          0.7381
Balanced Accuracy            0.59299             NA          0.5867
```

This model has a relatively high accuracy compared to other models. Also, this model has a relatively better performance in predicting negative system imbalances (i.e has a better precision)

Explainable Boosted Linear Regression

The second model used to tackle this problem is the use of linear regression and EBLR approach. First a linear regression model is trained using the feature set and some additional variables such as trend. Then, using the residuals obtained from the linear regression model, a decision tree is constructed. Then several nodes selected from the decision tree and added to the linear regression model as binary variables.

An additional feature is addded to the feature set which is called "last_obs" which represents the latest observation that will be available at the time of generating predictions. Thus, for the hours earlier than 12:00 lag 7 is used. For the remaining hours (after 12:00), observation at 05:00 is used.

Linear Regression Models

An iterative process is adopted to build the final linear regression model.

Trend

The first linear regression only included the ID as the trend variable. Its R-squared value is very low. Thus, it is not included in the final linear regression model

In [22]:
```
trend_lm <- lm(net~ID,train)
summary(trend_lm)
```

```
Call:
lm(formula = net ~ ID, data = train)

Residuals:
    Min      1Q  Median      3Q     Max 
-4112.1  -526.9   -81.1   502.1  8103.4 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) 6.555e+01  1.056e+01   6.205 5.55e-10 ***
ID          1.616e-02  7.134e-04  22.648  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 833.3 on 25390 degrees of freedom
Multiple R-squared:  0.0198,    Adjusted R-squared:  0.01976 
F-statistic: 512.9 on 1 and 25390 DF,  p-value: < 2.2e-16
```

Day of Week

Then, day of week information is added to the model. Two different versions are considered. First version included features such as is_sunday or is_weekday. The second version included day of week column as a factor in the linear regression model. Results and the plot of the first version are as follows:

In [23]:
```
dayoftheweek_lm <- lm(net ~ is_monday+is_wday+is_friday+is_saturday+is_sunday, tr
summary(dayoftheweek_lm)
```

```
Call:
lm(formula = net ~ is_monday + is_wday + is_friday + is_saturday +
    is_sunday, data = train)

Residuals:
    Min      1Q  Median      3Q     Max
-4166.8  -532.2   -61.5   494.3  8196.6

Coefficients: (1 not defined because of singularities)
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    58.49      13.89   4.210 2.56e-05 ***
is_monday     247.41      19.65  12.591  < 2e-16 ***
is_wday       283.74      16.04  17.691  < 2e-16 ***
is_friday     217.55      19.65  11.072  < 2e-16 ***
is_saturday   187.74      19.65   9.555  < 2e-16 ***
is_sunday         NA         NA      NA       NA
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 836.4 on 25387 degrees of freedom
Multiple R-squared:  0.01254,   Adjusted R-squared:  0.01238
F-statistic: 80.59 on 4 and 25387 DF,  p-value: < 2.2e-16
```

Results of the second version are as follows:

In [24]:
```
dayoftheweek2_lm <- lm(net ~ weekday, train)
summary(dayoftheweek2_lm)
```

```
Call:
lm(formula = net ~ weekday, data = train)

Residuals:
    Min      1Q  Median      3Q     Max
-4215.3  -530.3   -85.0   503.0  8265.4

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   205.39      11.80  17.403  < 2e-16 ***
weekday        17.01       2.64   6.443 1.19e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 841 on 25390 degrees of freedom
Multiple R-squared:  0.001632,  Adjusted R-squared:  0.001593
F-statistic: 41.51 on 1 and 25390 DF,  p-value: 1.194e-10
```

Since the first version produced better results in terms of R-squared, day of week information is included as features in the final linear regression model.

Hour Information

Then, hour information is added to the model. Two different versions are considered. First version included features such as is_over (hour equals 17,18,19, or 20). The second version included hour column as a factor in the linear regression model. Results of the first version are as follows:

In [24]:
```
dayoftheweek2_lm <- lm(net ~ weekday, train)
summary(dayoftheweek2_lm)
```

```
Call:
lm(formula = net ~ weekday, data = train)

Residuals:
    Min      1Q  Median      3Q     Max 
-4215.3  -530.3   -85.0   503.0  8265.4 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   205.39      11.80  17.403  < 2e-16 ***
weekday        17.01       2.64   6.443 1.19e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 841 on 25390 degrees of freedom
Multiple R-squared:  0.001632,  Adjusted R-squared:  0.001593 
F-statistic: 41.51 on 1 and 25390 DF,  p-value: 1.194e-10
```

Results and the plot of the second version are as follows:

In [26]:
```
hour2_lm <- lm(net ~ hour, train)
summary(hour2_lm)
```

```
Call:
lm(formula = net ~ hour, data = train)

Residuals:
    Min      1Q  Median      3Q     Max 
-4345.1  -513.7   -40.5   493.2  8083.1 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   102.52      25.41   4.036 5.46e-05 ***
hour1         135.93      35.93   3.783 0.000155 ***
hour2          67.71      35.93   1.884 0.059512 .  
hour3          16.10      35.93   0.448 0.654161    
hour4         -36.34      35.93  -1.012 0.311785    
hour5         -62.04      35.93  -1.727 0.084225 .  
hour6        -151.79      35.93  -4.225 2.40e-05 ***
hour7        -110.16      35.93  -3.066 0.002172 ** 
hour8          17.71      35.93   0.493 0.622106    
hour9         249.69      35.93   6.950 3.75e-12 ***
hour10        233.34      35.93   6.495 8.48e-11 ***
hour11        354.38      35.93   9.863  < 2e-16 ***
hour12        198.75      35.93   5.532 3.20e-08 ***
hour13        136.77      35.93   3.807 0.000141 ***
hour14        364.23      35.93  10.138  < 2e-16 ***
hour15        370.83      35.93  10.321  < 2e-16 ***
hour16        313.45      35.93   8.724  < 2e-16 ***
hour17        353.20      35.93   9.831  < 2e-16 ***
hour18        317.43      35.93   8.835  < 2e-16 ***
hour19        308.67      35.93   8.591  < 2e-16 ***
hour20        290.73      35.93   8.092 6.14e-16 ***
hour21        324.03      35.93   9.019  < 2e-16 ***
hour22        225.88      35.93   6.287 3.29e-10 ***
hour23        182.57      35.93   5.081 3.77e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 826.4 on 25368 degrees of freedom
Multiple R-squared:  0.03687,   Adjusted R-squared:  0.036 
F-statistic: 42.23 on 23 and 25368 DF,  p-value: < 2.2e-16
```

Since the second version produced better results in terms of R-squared, hour information is included as a factor in the final linear regression model.

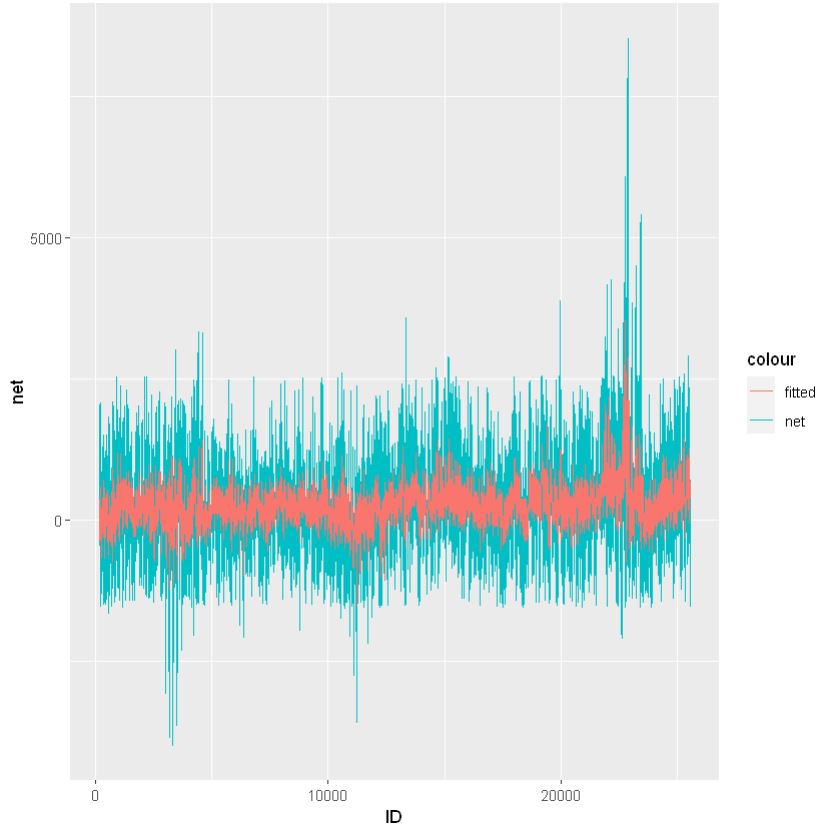
Lagged Variables

Summary of the linear regression with lagged variables and plot of its fitted values are as follows:

In [27]:
```
lag_lm <- lm(net ~ lag_24+lag_168+last_obs,train)
summary(lag_lm)
```

```
Call:
lm(formula = net ~ lag_24 + lag_168 + last_obs, data = train)

Residuals:
    Min      1Q  Median      3Q     Max 
-3770.8  -497.0   -35.6   465.5  6492.6 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) 1.558e+02  5.067e+00   30.75   <2e-16 ***
lag_24      1.897e-01  4.884e-03   38.85   <2e-16 ***
lag_168     1.918e-01  4.684e-03   40.94   <2e-16 ***
last_obs    1.668e-01  6.440e-03   25.90   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 764.1 on 25388 degrees of freedom
Multiple R-squared:  0.1759,    Adjusted R-squared:  0.1758 
F-statistic:  1807 on 3 and 25388 DF,  p-value: < 2.2e-16
```

In [95]:
```
ggplot(train, aes(x = ID))+
geom_line(aes(y = net, color = 'net'))+
geom_line(aes(y = predict(lag_lm,train), color = 'fitted'))
```



As you can see from the plot above, lagged variables are the best performing regressors in terms of explaining the variance in the data.

Combined Models

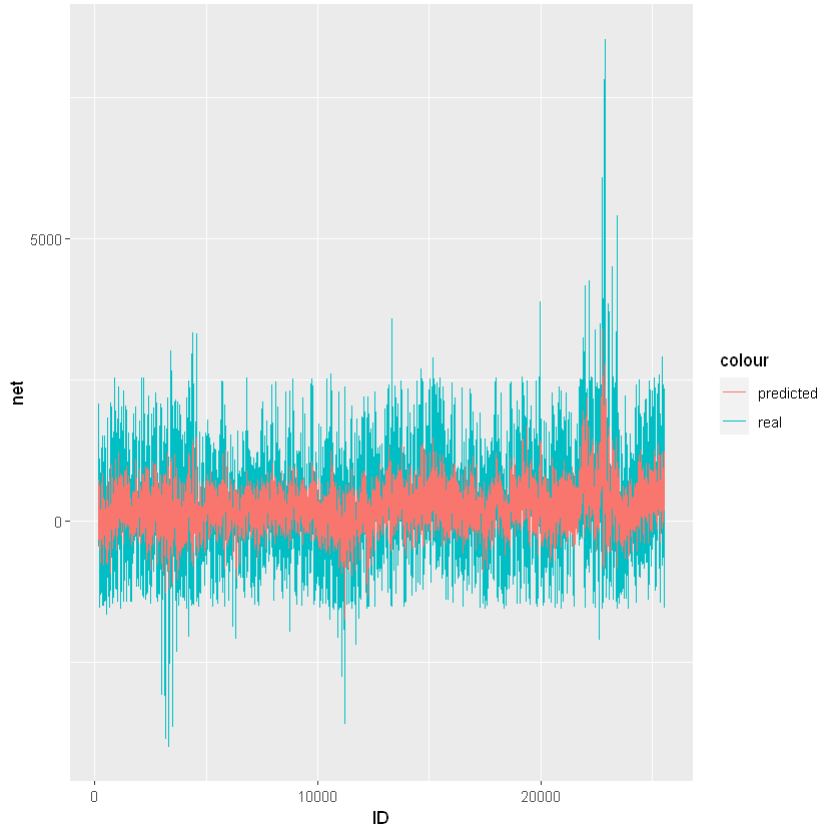
Finally, combined linear regression model is trained and its results are as follows:

In [48]:
```
comb1 <- lm(net ~ hour + is_monday + is_wday + is_friday + is_saturday + is_sunda
summary(comb1)
```

```
Call:
lm(formula = net ~ hour + is_monday + is_wday + is_friday + is_saturday + 
    is_sunday + lag_24 + lag_168 + last_obs, data = train)

Residuals:
    Min      1Q  Median      3Q     Max 
-3572.4  -488.7   -18.5   460.4  6377.9 

Coefficients: (1 not defined because of singularities)
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) -7.357e+01  2.589e+01  -2.842 0.004488 ** 
hour1        1.103e+02  3.274e+01   3.369 0.000755 ***
hour2        5.948e+01  3.273e+01   1.818 0.069148 .  
hour3       -2.801e+00  3.272e+01  -0.086 0.931767    
hour4       -8.690e+01  3.279e+01  -2.650 0.008050 ** 
hour5       -1.004e+02  3.279e+01  -3.063 0.002191 ** 
hour6       -1.790e+02  3.290e+01  -5.441 5.33e-08 ***
hour7       -7.171e+01  3.274e+01  -2.191 0.028492 *  
hour8       -2.362e+01  3.275e+01  -0.721 0.470735    
hour9        1.359e+02  3.276e+01   4.148 3.36e-05 ***
hour10       1.348e+02  3.275e+01   4.115 3.88e-05 ***
hour11       2.194e+02  3.280e+01   6.688 2.31e-11 ***
hour12       1.243e+02  3.274e+01   3.797 0.000147 ***
hour13       8.646e+01  3.273e+01   2.642 0.008250 ** 
hour14       2.259e+02  3.281e+01   6.885 5.89e-12 ***
hour15       2.361e+02  3.280e+01   7.197 6.31e-13 ***
hour16       2.044e+02  3.277e+01   6.237 4.52e-10 ***
hour17       2.190e+02  3.280e+01   6.676 2.51e-11 ***
hour18       1.881e+02  3.280e+01   5.737 9.76e-09 ***
hour19       1.798e+02  3.280e+01   5.483 4.23e-08 ***
hour20       1.618e+02  3.280e+01   4.932 8.18e-07 ***
hour21       1.830e+02  3.281e+01   5.579 2.45e-08 ***
hour22       1.240e+02  3.276e+01   3.784 0.000155 ***
hour23       1.063e+02  3.274e+01   3.248 0.001165 ** 
is_monday    2.778e+02  1.783e+01  15.585  < 2e-16 ***
is_wday      1.753e+02  1.455e+01  12.044  < 2e-16 ***
is_friday    1.057e+02  1.776e+01   5.951 2.69e-09 ***
is_saturday  1.041e+02  1.773e+01   5.870 4.41e-09 ***
is_sunday           NA         NA      NA       NA    
lag_24       1.715e-01  4.924e-03  34.822  < 2e-16 ***
lag_168      1.667e-01  4.708e-03  35.405  < 2e-16 ***
last_obs     1.997e-01  6.535e-03  30.561  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 752.4 on 25361 degrees of freedom
Multiple R-squared:  0.2017,    Adjusted R-squared:  0.2007 
F-statistic: 213.6 on 30 and 25361 DF,  p-value: < 2.2e-16
```

Plot of the fitted values are as follows:

In [99]:
```
ggplot(train, aes(x=ID))+
geom_line(aes(y=net, color="real"))+
geom_line(aes(y=predict(comb1,train), color = 'predicted'))
```

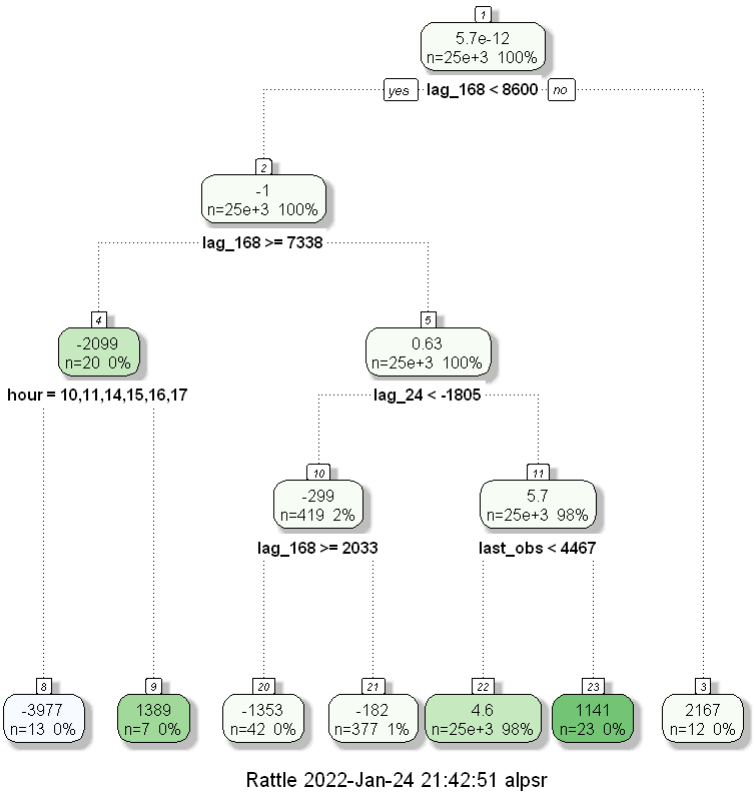


EBLR Approach

At this step, decision trees are constructed on the residuals of the fitted linear regression model. For simplification, maximum depth of the decision trees are set to 4. After constructing the decision trees, one or several binary variables are added to the model in order to further explain the residuals. This iterative process is continued until there are no further improvements to the linear regression model.

Results of the first decision tree on the residuals of the linear regression is as follows:

In [32]:
```
train[,Residuals1 := predict(comb1, train) - net]
tree1 = rpart(Residuals1 ~ hour + is_monday + is_wday + is_friday + is_saturday +
              train,control=rpart.control(cp=0,maxdepth=4))
fancyRpartPlot(tree1)
```

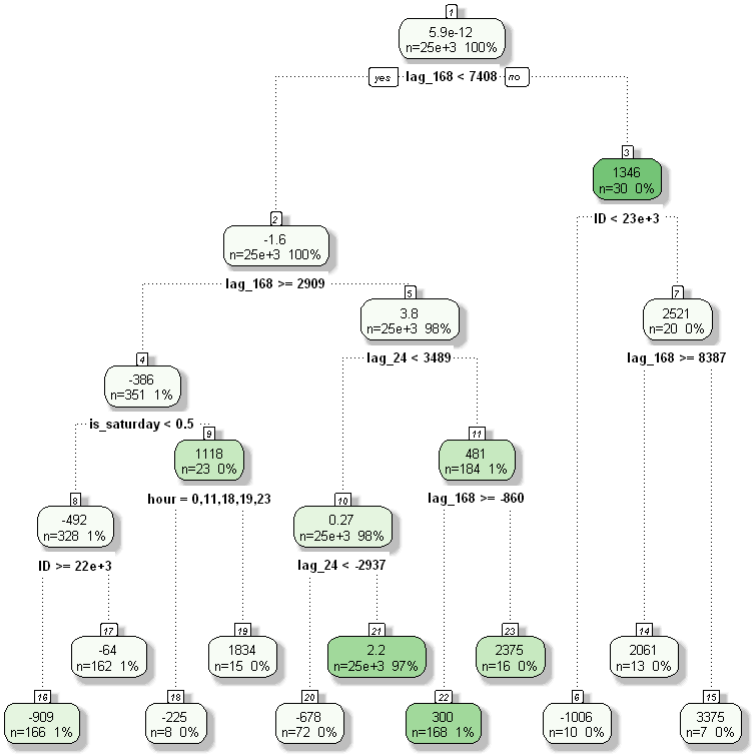


A binary variable which denotes the cases with lag_168 between 8600 and 7338 and hour in 10,11,14,15,16,17 are added to the model. Then, a new linear regression model is constructed. As a result, adjusted R-squared improved to 21.47%. Then a new decision tree is formed on the new residuals.

In [34]:
```
comb2 <- lm(net ~ ID + hour + is_monday + is_wday + is_friday + is_saturday +
            is_sunday + lag_24 + lag_168 + last_obs + Binary1, train)
summary(comb2)
```

```
Call:
lm(formula = net ~ ID + hour + is_monday + is_wday + is_friday +
    is_saturday + is_sunday + lag_24 + lag_168 + last_obs + Binary1,
    data = train)

Residuals:
    Min      1Q  Median      3Q     Max
-5892.8  -485.8   -19.5   461.2  5657.7

Coefficients: (1 not defined because of singularities)
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.477e+02  2.708e+01  -5.452 5.01e-08 ***
ID           5.549e-03  6.593e-04   8.417  < 2e-16 ***
hour1        1.137e+02  3.246e+01   3.502 0.000462 ***
hour2        6.112e+01  3.244e+01   1.884 0.059572 .
hour3       -2.002e+00  3.243e+01  -0.062 0.950786
hour4       -8.643e+01  3.250e+01  -2.659 0.007835 **
hour5       -1.008e+02  3.250e+01  -3.103 0.001919 **
hour6       -1.819e+02  3.261e+01  -5.578 2.46e-08 ***
hour7       -7.561e+01  3.245e+01  -2.330 0.019824 *
hour8       -2.273e+01  3.246e+01  -0.700 0.483775
hour9        1.433e+02  3.247e+01   4.412 1.03e-05 ***
hour10       1.338e+02  3.247e+01   4.122 3.76e-05 ***
hour11       2.220e+02  3.252e+01   6.826 8.95e-12 ***
hour12       1.298e+02  3.245e+01   4.001 6.33e-05 ***
hour13       9.010e+01  3.244e+01   2.777 0.005484 **
hour14       2.288e+02  3.253e+01   7.034 2.06e-12 ***
hour15       2.348e+02  3.252e+01   7.220 5.34e-13 ***
hour16       2.050e+02  3.249e+01   6.309 2.86e-10 ***
hour17       2.215e+02  3.252e+01   6.812 9.87e-12 ***
hour18       1.980e+02  3.251e+01   6.090 1.14e-09 ***
hour19       1.897e+02  3.251e+01   5.833 5.51e-09 ***
hour20       1.716e+02  3.251e+01   5.278 1.32e-07 ***
hour21       1.939e+02  3.253e+01   5.959 2.57e-09 ***
hour22       1.317e+02  3.248e+01   4.054 5.06e-05 ***
hour23       1.120e+02  3.246e+01   3.451 0.000560 ***
is_monday    2.790e+02  1.767e+01  15.787  < 2e-16 ***
is_wday      1.815e+02  1.443e+01  12.572  < 2e-16 ***
is_friday    1.148e+02  1.762e+01   6.514 7.44e-11 ***
is_saturday  1.102e+02  1.757e+01   6.270 3.67e-10 ***
is_sunday           NA         NA      NA       NA
lag_24       1.661e-01  4.922e-03  33.746  < 2e-16 ***
lag_168      1.452e-01  4.790e-03  30.311  < 2e-16 ***
last_obs     1.949e-01  6.497e-03  29.994  < 2e-16 ***
Binary1      4.100e+03  2.100e+02  19.519  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 745.9 on 25359 degrees of freedom
Multiple R-squared:  0.2157,    Adjusted R-squared:  0.2147
F-statistic: 217.9 on 32 and 25359 DF,  p-value: < 2.2e-16
```

In [35]:
```
train[,Residuals2 := predict(comb2, train) - net]
tree2 = rpart(Residuals2 ~ ID + hour + is_monday + is_wday + is_friday +
              is_saturday + is_sunday + lag_24 + lag_168 + last_obs + Binary1,
              train,control=rpart.control(cp=0,maxdepth=4))
fancyRpartPlot(tree2)
```



Rattle 2022-Jan-24 21:42:52 alpsr

Three new binary variables are added to the model corresponding to the cases with highest average residuals. Then, a new model is constructed. The resulting model's adjusted R-squared value improved to 21.98%.

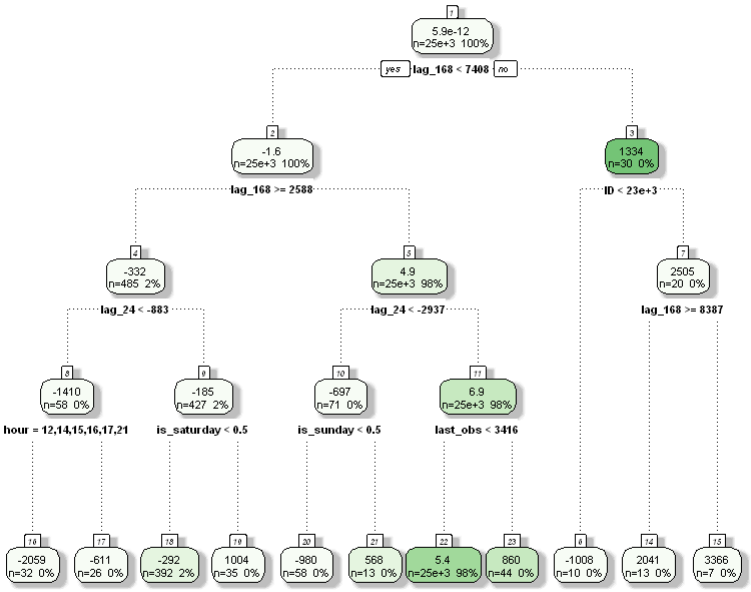
In [37]:
```
comb3 <- lm(net ~ ID + hour + is_monday + is_wday + is_friday + is_saturday +
            is_sunday + lag_24 + lag_168 + last_obs + Binary1 + Binary2 + Binary3
summary(comb3)
```

```
Call:
lm(formula = net ~ ID + hour + is_monday + is_wday + is_friday + 
    is_saturday + is_sunday + lag_24 + lag_168 + last_obs + Binary1 + 
    Binary2 + Binary3 + Binary4, data = train)

Residuals:
    Min      1Q  Median      3Q     Max 
-5904.9  -486.6   -20.4   460.2  5670.8 

Coefficients: (3 not defined because of singularities)
              Estimate Std. Error t value Pr(>|t|)    
(Intercept) -1.521e+02  2.700e+01  -5.635 1.77e-08 ***
ID           5.701e-03  6.573e-04   8.674  < 2e-16 ***
hour1        1.133e+02  3.235e+01   3.501 0.000465 ***
hour2        6.091e+01  3.233e+01   1.884 0.059598 .  
hour3       -1.989e+00  3.233e+01  -0.062 0.950938    
hour4       -8.615e+01  3.240e+01  -2.659 0.007836 ** 
hour5       -1.005e+02  3.240e+01  -3.102 0.001923 ** 
hour6       -1.812e+02  3.251e+01  -5.574 2.51e-08 ***
hour7       -7.514e+01  3.235e+01  -2.323 0.020184 *  
hour8       -2.260e+01  3.235e+01  -0.699 0.484775    
hour9        1.427e+02  3.237e+01   4.408 1.05e-05 ***
hour10       1.332e+02  3.236e+01   4.116 3.87e-05 ***
hour11       2.232e+02  3.242e+01   6.886 5.88e-12 ***
hour12       1.339e+02  3.235e+01   4.138 3.51e-05 ***
hour13       9.433e+01  3.234e+01   2.917 0.003534 ** 
hour14       2.300e+02  3.242e+01   7.095 1.33e-12 ***
hour15       2.383e+02  3.242e+01   7.352 2.01e-13 ***
hour16       2.087e+02  3.238e+01   6.445 1.18e-10 ***
hour17       2.250e+02  3.242e+01   6.942 3.95e-12 ***
hour18       2.016e+02  3.241e+01   6.221 5.01e-10 ***
hour19       1.910e+02  3.241e+01   5.893 3.84e-09 ***
hour20       1.729e+02  3.241e+01   5.336 9.61e-08 ***
hour21       1.928e+02  3.243e+01   5.946 2.78e-09 ***
hour22       1.309e+02  3.237e+01   4.044 5.27e-05 ***
hour23       1.115e+02  3.235e+01   3.445 0.000571 ***
is_monday    2.816e+02  1.762e+01  15.986  < 2e-16 ***
is_wday      1.825e+02  1.439e+01  12.687  < 2e-16 ***
is_friday    1.176e+02  1.756e+01   6.696 2.18e-11 ***
is_saturday  1.171e+02  1.753e+01   6.679 2.46e-11 ***
is_sunday           NA         NA      NA       NA    
lag_24       1.725e-01  4.931e-03  34.991  < 2e-16 ***
lag_168      1.414e-01  4.783e-03  29.565  < 2e-16 ***
last_obs     1.944e-01  6.476e-03  30.022  < 2e-16 ***
Binary1      4.118e+03  2.094e+02  19.667  < 2e-16 ***
Binary2     -2.416e+03  1.875e+02 -12.888  < 2e-16 ***
Binary3             NA         NA      NA       NA    
Binary4             NA         NA      NA       NA    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 743.4 on 25358 degrees of freedom
Multiple R-squared:  0.2208,    Adjusted R-squared:  0.2198
F-statistic: 217.7 on 33 and 25358 DF,  p-value: < 2.2e-16
```

Two new binary variables are created and a new model is constructed. Adjusted R-squared increased to 22.52%. Since the improvement in the model is relatively small, no further iterations are made. Plot of the fitted values of the model is as follows:

In [38]:
```
fancyRpartPlot(tree3)
```



Rattle 2022-Jan-24 21:42:53 alpsr

In conclusion, six binary variables are added to the model with the use of decision trees. Summary of the model is displayed below.

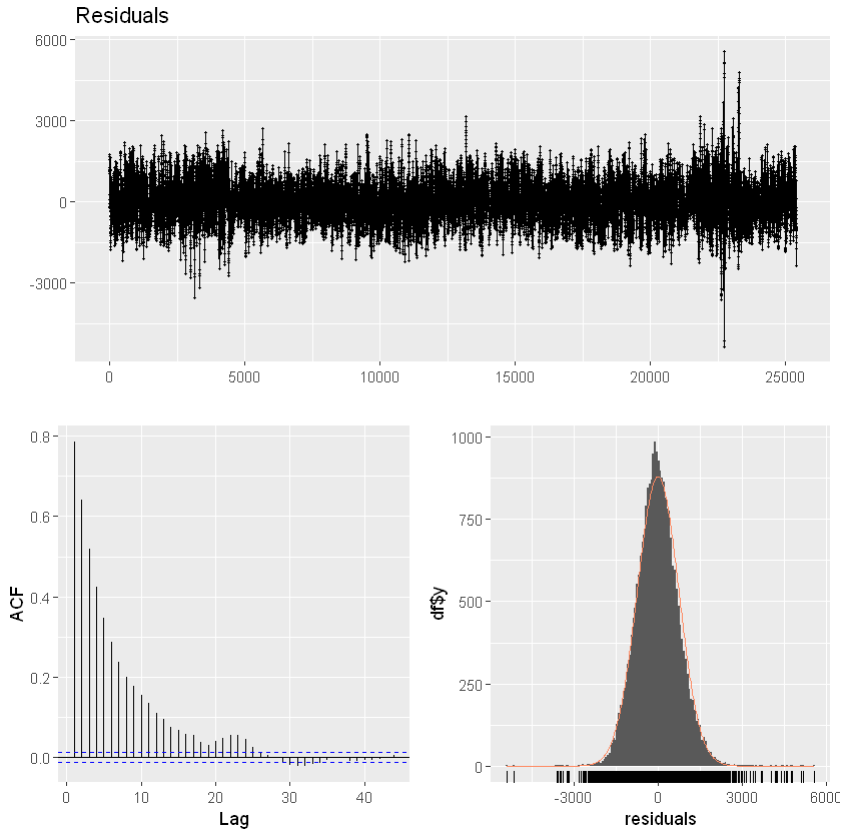
In [39]:
```
summary(comb4)
```

```
Call:
lm(formula = net ~ ID + hour + is_monday + is_wday + is_friday + 
    is_saturday + is_sunday + lag_24 + lag_168 + last_obs + Binary1 + 
    Binary2 + Binary3 + Binary4 + Binary5 + Binary6, data = train)

Residuals:
    Min      1Q  Median      3Q     Max 
-5359.3  -486.7   -19.4   458.8  5576.1 

Coefficients: (3 not defined because of singularities)
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) -1.507e+02  2.690e+01  -5.600 2.17e-08 ***
ID           5.680e-03  6.549e-04   8.673  < 2e-16 ***
hour1        1.115e+02  3.224e+01   3.460 0.000542 ***
hour2        6.001e+01  3.222e+01   1.863 0.062525 .  
hour3       -2.290e+00  3.221e+01  -0.071 0.943334    
hour4       -8.594e+01  3.228e+01  -2.662 0.007770 ** 
hour5       -9.990e+01  3.228e+01  -3.095 0.001973 ** 
hour6       -1.792e+02  3.239e+01  -5.534 3.17e-08 ***
hour7       -7.327e+01  3.223e+01  -2.273 0.023032 *  
hour8       -2.275e+01  3.224e+01  -0.706 0.480499    
hour9        1.394e+02  3.226e+01   4.323 1.55e-05 ***
hour10       1.301e+02  3.225e+01   4.035 5.48e-05 ***
hour11       2.214e+02  3.230e+01   6.853 7.40e-12 ***
hour12       1.343e+02  3.224e+01   4.165 3.12e-05 ***
hour13       9.731e+01  3.222e+01   3.020 0.002530 ** 
hour14       2.297e+02  3.231e+01   7.111 1.18e-12 ***
hour15       2.375e+02  3.230e+01   7.353 2.00e-13 ***
hour16       2.082e+02  3.227e+01   6.453 1.12e-10 ***
hour17       2.237e+02  3.230e+01   6.924 4.50e-12 ***
hour18       2.030e+02  3.230e+01   6.284 3.35e-10 ***
hour19       1.894e+02  3.230e+01   5.864 4.58e-09 ***
hour20       1.701e+02  3.229e+01   5.266 1.41e-07 ***
hour21       1.895e+02  3.231e+01   5.865 4.55e-09 ***
hour22       1.273e+02  3.226e+01   3.947 7.93e-05 ***
hour23       1.088e+02  3.224e+01   3.375 0.000738 ***
is_monday    2.781e+02  1.756e+01  15.839  < 2e-16 ***
is_wday      1.793e+02  1.434e+01  12.507  < 2e-16 ***
is_friday    1.185e+02  1.750e+01   6.772 1.29e-11 ***
is_saturday  1.141e+02  1.747e+01   6.533 6.55e-11 ***
is_sunday           NA         NA      NA       NA    
lag_24       1.722e-01  4.914e-03  35.040  < 2e-16 ***
lag_168      1.542e-01  4.880e-03  31.597  < 2e-16 ***
last_obs     1.953e-01  6.454e-03  30.257  < 2e-16 ***
Binary1      6.622e+03  2.875e+02  23.036  < 2e-16 ***
Binary2     -2.396e+03  1.868e+02 -12.825  < 2e-16 ***
Binary3             NA         NA      NA       NA    
Binary4             NA         NA      NA       NA    
Binary5     -3.152e+03  2.729e+02 -11.549  < 2e-16 ***
Binary6     -1.820e+03  1.972e+02  -9.230  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

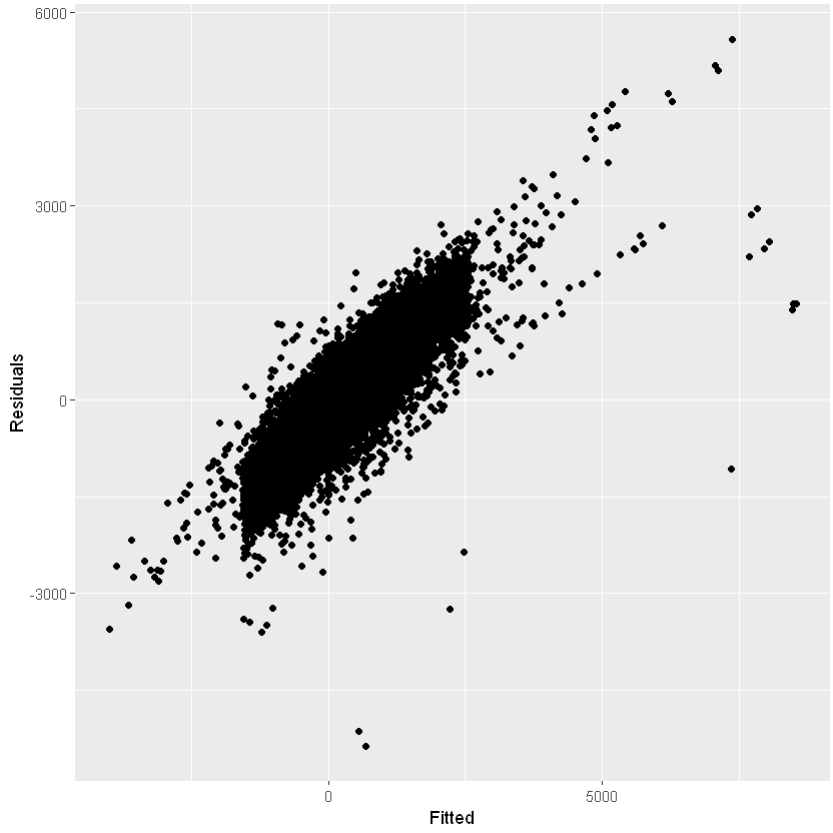
```
Residual standard error: 740.8 on 25356 degrees of freedom
Multiple R-squared:  0.2263,    Adjusted R-squared:  0.2252
F-statistic: 211.9 on 35 and 25356 DF,  p-value: < 2.2e-16
```

Analysis of the residuals show that there is a high autocorrelation in the residuals which means that there are some unexplained relationships between the residuals. Additional features are needed to account for these relationships. On the other hand, based on the visual analysis of the plot and the histogram, residuals seemed to follow a white noise pattern and normal distribution with a mean around zero.

In [40]:
```
checkresiduals(comb4$residuals)
```



In [61]:
```
ggplot(train, aes(x=net))+
geom_point(aes(y = comb4$residuals))+
labs(y = "Residuals")+
labs(x = "Fitted")
```



The model has a very large WMAPE value which means that the predictions from the model may not be reliable. Then, the outputs of the model are used to make a classification over the test period with a sliding window approach, the following confusion matrix is obtained. Although the model has a high accuracy with approximately 70%, it fails to account for the negative cases and

always predicts positive values which means there is a significant bias in the model. Thus, the model achieves this high accuracy through the unbalanced nature of the data set and has a very low precision.

In [41]:
```
train[, Final_Predict := predict(comb4, train)]
perf_dt("EBLR",train$net,train$Final_Predict)
```

A data.frame: 1 × 9

name	n	mean	sd	FBias	MAPE	RMSE	MAD	WMAPE
<chr>	<int>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
EBLR	25392	273.4027	841.6478	-2.157312e-14	Inf	4.645743	577.0092	2.110474

In [52]:
```
confusionMatrix(data = as.factor(est_directions), reference = as.factor(real_dire
```

```
Confusion Matrix and Statistics

          Reference
Prediction  -1   0   1
        -1   0   0   0
        0    0   0   0
        1   37  11 120

Overall Statistics

               Accuracy : 0.7143
                 95% CI : (0.6396, 0.7812)
    No Information Rate : 0.7143
    P-Value [Acc > NIR] : 0.5388

                  Kappa : 0

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: -1 Class: 0 Class: 1
Precision                   NA       NA   0.7143
Recall                  0.0000  0.00000   1.0000
F1                          NA       NA   0.8333
Prevalence              0.2202  0.06548   0.7143
Detection Rate          0.0000  0.00000   0.7143
Detection Prevalence    0.0000  0.00000   1.0000
Balanced Accuracy       0.5000  0.50000   0.5000
```

KNN with different distance calculations

In this approach, the main objective is to determine which distance calculation gives the best accuracy performance in the dataset. One of the problem in this model is to calculation of the distance matrix. Computationally it is impossible to get a distance matrix for all instances. So, a

train set is selected to get performance of the K-Nearest Neighbor algorithm. To make it realistic train set is selected before determined test starting day (2021-12-01). In addition, 14 days is selected to get a reasonable size and feasible computation range. As mentioned before, the main aim is to determine whether difference distance calculation has any impact on the model is the major objective to analysis. Weather dataset with principal components, day, and lagged values are used in the input dataset. Tried distance calculations are as follows:

- Euclidean distance when k=2,5,10
- DTW distance + no window when k=2,5,10
- DTW distance + window.type='sakoechiba' + window.size=10 when k=2,5,10
- DTW distance + window.type='sakoechiba' + window.size=20 when k=2,5,10
- LCSS + epsilon=0.05 + no window when k=2,5,10
- LCSS + epsilon=0.1 + no window when k=2,5,10
- ERP + gap penalty=0.5 + no window when k=2,5,10
- ERP + gap penalty=1 + no window when k=2,5,10

In addition, "Cross-validation to evaluate alternative classification strategies" Notebook is used to get these performance values. Namely, the performance of the models has been evaluated with cross-validation with fixed test indices to get better comparison results.

Obtained results

In [33]:
```
overall_results=rbindlist(result)
overall_results[,list(avg_acc=mean(acc),sdev_acc=sd(acc),result_count=.N),by=list
```

approach	k	avg_acc	sdev_acc	result_count
project_dtw_raw_dist.csv	2	0.6430769	0.06964976	100
project_dtw_raw_dist.csv	5	0.6802294	0.05341675	100
project_dtw_raw_dist.csv	10	0.7200742	0.03903751	100
project_dtw_raw_dist_sakoe_10.csv	2	0.6430769	0.06964976	100
project_dtw_raw_dist_sakoe_10.csv	5	0.6802294	0.05341675	100
project_dtw_raw_dist_sakoe_10.csv	10	0.7200742	0.03903751	100
project_dtw_raw_dist_sakoe_20.csv	2	0.6430769	0.06964976	100
project_dtw_raw_dist_sakoe_20.csv	5	0.6802294	0.05341675	100
project_dtw_raw_dist_sakoe_20.csv	10	0.7200742	0.03903751	100
project_erp_raw_gap_005.csv	2	0.6399798	0.06650581	100
project_erp_raw_gap_005.csv	5	0.6909312	0.04906949	100
project_erp_raw_gap_005.csv	10	0.7182456	0.04072530	100
project_erp_raw_gap_1.csv	2	0.6402227	0.06567888	100
project_erp_raw_gap_1.csv	5	0.6846829	0.04848222	100
project_erp_raw_gap_1.csv	10	0.7198178	0.03875830	100
project_euc_raw_dist.csv	2	0.6328745	0.06607764	100
project_euc_raw_dist.csv	5	0.6764035	0.05660061	100
project_euc_raw_dist.csv	10	0.7133131	0.04276158	100
project_lcss_raw_epsilon_005.csv	2	0.6087045	0.05901520	100
project_lcss_raw_epsilon_005.csv	5	0.7085290	0.02832335	100
project_lcss_raw_epsilon_005.csv	10	0.7319771	0.01909082	100
project_lcss_raw_epsilon_01.csv	2	0.6241430	0.05959275	100
project_lcss_raw_epsilon_01.csv	5	0.7114440	0.03103907	100
project_lcss_raw_epsilon_01.csv	10	0.7333198	0.01551811	100

The best 8 distance types

In [69]:
```
head(res_dt[order(avg_acc,decreasing = TRUE)], 8)
```

approach	k	avg_acc	sdev_acc	result_count
project_lcss_raw_epsilon_01.csv	10	0.7333198	0.01551811	100
project_lcss_raw_epsilon_005.csv	10	0.7319771	0.01909082	100
project_dtw_raw_dist.csv	10	0.7200742	0.03903751	100
project_dtw_raw_dist_sakoe_10.csv	10	0.7200742	0.03903751	100
project_dtw_raw_dist_sakoe_20.csv	10	0.7200742	0.03903751	100
project_erp_raw_gap_1.csv	10	0.7198178	0.03875830	100
project_erp_raw_gap_005.csv	10	0.7182456	0.04072530	100
project_euc_raw_dist.csv	10	0.7133131	0.04276158	100

By looking this table, it can be said that there is no huge difference in between the performance of the models. In addition, calculation of distance matrix is much faster than others, which directed us to use euclidean distance calculations in the KNN models. K is selected 10 for the predictions by looking at the best models' k value.

Performance of the KNN by using instances at 2021

In [190]:
```
confusionMatrix(data = as.factor(class_predictions$predicted), reference = as.fac
```

```
Confusion Matrix and Statistics

          Reference
Prediction Negative Neutral Positive
  Negative       40       7       30
  Neutral         2       5        2
  Positive       28      17      205

Overall Statistics

               Accuracy : 0.744
                 95% CI : (0.6939, 0.7899)
    No Information Rate : 0.7054
    P-Value [Acc > NIR] : 0.065910

                  Kappa : 0.3979

 Mcnemar's Test P-Value : 0.002103

Statistics by Class:

                     Class: Negative Class: Neutral Class: Positive
Precision                     0.5195        0.55556          0.8200
Recall                        0.5714        0.17241          0.8650
F1                            0.5442        0.26316          0.8419
Prevalence                    0.2083        0.08631          0.7054
Detection Rate                0.1190        0.01488          0.6101
Detection Prevalence          0.2292        0.02679          0.7440
Balanced Accuracy             0.7162        0.57969          0.7052
```

Obtained accuracy value is 0.744 with this approach. One of the problems in this problem is having an imbalanced target variable, which causes a tendency to predict positive. In order to overcome this problem, an alternative approach is tried to get better performance in the model.

KNN with Sax Representation

As mentioned in previous approach, an alternative approach to overcome this problem is tried. By knowing outliers can disturb the model, sax representation will be used. In addition, 10 different class with equal probability will be created by using cut function to overcome imbalance problem in target variable. Class information and their actual values as follows:

- 1, 2, 3 are Negative Class (30%)
- 4 is Neutral Class (10%)
- 5, 6, 7, 8, 9, 10 are Positive Class (60%)

After predicting these 10 classes a conversion process will be applied to get actual target variables.

Obtained Result with 10 Classes

In [62]:
```
table(actual=res_dt$actual,  predicted=res_dt$predicted)
```

```
      predicted
actual 1 2 3 4 5 6 7 8 9 10
    1  0 2 1 1 0 0 1 1 2  1
    2  2 2 2 3 1 0 2 3 1  0
    3  3 3 1 7 2 4 1 1 4  1
    4  6 5 6 6 4 6 7 4 3  0
    5  3 3 8 4 1 1 1 1 1  2
    6  3 8 7 8 2 4 4 0 0  1
    7  1 2 5 7 3 5 3 3 5  1
    8  7 6 8 5 5 4 6 4 2  4
    9  5 4 6 9 2 8 4 6 4  3
    10 0 0 2 6 1 2 9 3 6  9
```

It is hard to understand the performance of the model by looking at this table. In addition, it can be misleading when converted actual target variable type. So, the converted version must be controlled. Predictions after conversion are as follows:

Obtained Result with 10 Classes after Conversion

In [37]:
```
confusionMatrix(data = as.factor(res_dt$converted_pred), reference = as.factor(re
```

```
Confusion Matrix and Statistics

          Reference
Prediction Negative Neutral Positive
  Negative       28       8       83
  Neutral        15       6       40
  Positive       27      15      114

Overall Statistics

               Accuracy : 0.4405
                 95% CI : (0.3866, 0.4954)
    No Information Rate : 0.7054
    P-Value [Acc > NIR] : 1

                  Kappa : 0.0404

 Mcnemar's Test P-Value : 4.006e-09

Statistics by Class:

                     Class: Negative Class: Neutral Class: Positive
Precision                    0.23529        0.09836          0.7308
Recall                       0.40000        0.20690          0.4810
F1                           0.29630        0.13333          0.5802
Prevalence                   0.20833        0.08631          0.7054
Detection Rate               0.08333        0.01786          0.3393
Detection Prevalence         0.35417        0.18155          0.4643
Balanced Accuracy            0.52895        0.51387          0.5284
```

There is a certain performance decrease, so this approach will be ignored. Much as problem imbalance tried to be handled, decrease in performance is too big to neglect. One of the reasons for the decrease in performance can be explained by insufficient feature size to distinguish classes.

Conclusion

In this project, the aim was to predict the sign of the imbalance in the Turkish Electricity Markets. Several classification and regression models are constructed. In the regression, first the "net imbalance" is forecasted, then the sign of the forecast is used for classification. Time series information about the past imbalances, the past meteorological measurements and breakdown data to create features. Then, performance of these models are evaluated over the test period. The performance of these models are considerable affected by the imbalanced nature of the target variable. In other words, "Positive" labeled instances comprised the majority of the data set which resulted in low precision for the "Negative" labeled cases.

Numerous approaches are considered for the classification problem which was explained in the previous sections. No single model has performed significantly better than the other models. We have achieved a training accuracy of 73.33% with k-NN classifier on LCSS distance with epsilon =

0.1 adn k = 10 on the raw time series. On the other hand, we have achieved an accuracy of approximately 70% with the k-NN classifier using the windowed time series and node information created using the decision trees. Also, with this approach we have achieved a relatively better precision on the negative values. However, since there are multiple models which resulted in similar accuracies, while making our predictions, we have considered several of them at the same time. We have also tried to adress the imbalance problem with the use of SAX representation which divides the data into equal probability classes.

Possible extensions to this study includes the addition of the past forecasted weather measurements. With this data we can compared the deviations of the real weather measurements from the predicted measurements which can then be used to explain the errors in the electricity production and consumption forecasts. These errors result in the imbalances in the electricity markets. Also, we can use the oversampling and undersampling methods to counter the imbalanced nature of the data set. Finally, more advanced algorithms such as random forests, XGBoosting and SVM can be used to construct more accurate models. In conclusion, although there is still room for improvement, we have achieved to increase the baseline method's accuracy to approximately 70%.

Codes

Repository of Project: https://github.com/mbahadir/IE48B_project_files (https://github.com/mbahadir/IE48B_project_files)

Breakdown Data Notebook:
https://github.com/mbahadir/IE48B_project_files/blob/main/Breakdown_Data.ipynb
(https://github.com/mbahadir/IE48B_project_files/blob/main/Breakdown_Data.ipynb)
Breakdown Data R File:
https://github.com/mbahadir/IE48B_project_files/blob/main/Breakdown_Data.r
(https://github.com/mbahadir/IE48B_project_files/blob/main/Breakdown_Data.r)

Data Control in Python Notebook:
https://github.com/mbahadir/IE48B_project_files/blob/main/Data%20Control%20in%20Python.ipynb
(https://github.com/mbahadir/IE48B_project_files/blob/main/Data%20Control%20in%20Python.ipynb
Data Control in Python File:
https://github.com/mbahadir/IE48B_project_files/blob/main/Data%20Control%20in%20Python.py
(https://github.com/mbahadir/IE48B_project_files/blob/main/Data%20Control%20in%20Python.py)

Data Control in R Notebook:
https://github.com/mbahadir/IE48B_project_files/blob/main/Data%20Control%20in%20R.ipynb
(https://github.com/mbahadir/IE48B_project_files/blob/main/Data%20Control%20in%20R.ipynb)
Data Control in R File:
https://github.com/mbahadir/IE48B_project_files/blob/main/Data%20Control%20in%20R.r
(https://github.com/mbahadir/IE48B_project_files/blob/main/Data%20Control%20in%20R.r)

KNN Files Notebook:
https://github.com/mbahadir/IE48B_project_files/blob/main/KNN%20Files.ipynb
(https://github.com/mbahadir/IE48B_project_files/blob/main/KNN%20Files.ipynb)

KNN Files R File: https://github.com/mbahadir/IE48B_project_files/blob/main/KNN%20Files.r
(https://github.com/mbahadir/IE48B_project_files/blob/main/KNN%20Files.r)

Sax Representation with KNN Notebook:
https://github.com/mbahadir/IE48B_project_files/blob/main/Sax%20Representation%20with%20KNN
(https://github.com/mbahadir/IE48B_project_files/blob/main/Sax%20Representation%20with%20KNI
Sax Representation with KNN R File:
https://github.com/mbahadir/IE48B_project_files/blob/main/Sax%20Representation%20with%20KNN
(https://github.com/mbahadir/IE48B_project_files/blob/main/Sax%20Representation%20with%20KNI

Linear Regression Final Notebook:
https://github.com/mbahadir/IE48B_project_files/blob/main/Linear%20Regression%20Final.ipynb
(https://github.com/mbahadir/IE48B_project_files/blob/main/Linear%20Regression%20Final.ipynb)
Linear Regression Final R File:
https://github.com/mbahadir/IE48B_project_files/blob/main/Linear%20Regression%20Final.r
(https://github.com/mbahadir/IE48B_project_files/blob/main/Linear%20Regression%20Final.r)

Decision Tree - Final Notebook:
https://github.com/mbahadir/IE48B_project_files/blob/main/Decision%20Tree%20-%20Final.ipynb
(https://github.com/mbahadir/IE48B_project_files/blob/main/Decision%20Tree%20-%20Final.ipynb)
Decision Tree - Final R File:
https://github.com/mbahadir/IE48B_project_files/blob/main/Decision%20Tree%20-%20Final.r
(https://github.com/mbahadir/IE48B_project_files/blob/main/Decision%20Tree%20-%20Final.r)

Weather Data Set Notebook:
https://github.com/mbahadir/IE48B_project_files/blob/main/Weather%20Data%20Set.ipynb
(https://github.com/mbahadir/IE48B_project_files/blob/main/Weather%20Data%20Set.ipynb)
Weather Data Set R File:
https://github.com/mbahadir/IE48B_project_files/blob/main/Weather%20Data%20Set.r
(https://github.com/mbahadir/IE48B_project_files/blob/main/Weather%20Data%20Set.r)