MEF UNIVERSITY

CARBON PRICE FORECASTING

Capstone Project

Nurhak Karakaya

İSTANBUL, 2018

MEF UNIVERSITY

CARBON PRICE FORECASTING

Capstone Project

Nurhak Karakaya

Advisor: Prof. Semra Ağralı

İSTANBUL, 2018

MEF UNIVERSITY

Name of the project: Carbon Price Forecasting
Name/Last Name of the Student: Nurhak Karakaya
Date of Thesis Defense: 10/08/2018

I hereby state that the graduation project prepared by Nurhak Karakaya has been completed under my supervision. I accept this work as a "Graduation Project".

10/08/2018
Prof. Semra Ağralı

I hereby state that I have examined this graduation project by Nurhak Karakaya which is accepted by his supervisor. This work is acceptable as a graduation project and the student is eligible to take the graduation project examination.

10/08/2018

Prof. Özgür Özlük
Director
of
Big Data Analytics Program

We hereby state that we have held the graduation examination of _____ and agree that the student has satisfied all requirements.

THE EXAMINATION COMMITTEE

Committee Member	Signature
1. Prof. Semra Ağralı	……..
2. ……..	……..

Academic Honesty Pledge

I promise not to collaborate with anyone, not to seek or accept any outside help, and not to give any help to others.

I understand that all resources in print or on the web must be explicitly cited.

In keeping with MEF University's ideals, I pledge that this work is my own and that I have neither given nor received inappropriate assistance in preparing it.

Name Date Signature

EXECUTIVE SUMMARY

CARBON PRICE FORECASTING

Nurhak Karakaya

Advisor: Prof. Semra Ağralı

AUGUST, 2018, 88 pages

In last twenty years great improvements occurred both in technological advances and in the world economic capacity. The total production capacity of countries has been increasing rapidly. These increases need great usage of energy. For that reason, prices of energy related products are very important as they dramatically affect company budgets. Energy budgets get a great deal in total budget of companies and countries. A unit increase in an energy related product can severely affect the budget. The carbon price is one of those products. Besides carbon prices, carbon usage also affects global environment so its price also has an impact on global temperature.

To forecast future carbon price different machine learning methods are used. In literature, support vector machines (SVM) [1, 2, 3], random forest (RF) [4, 5], artificial neural networks (ANN) [6, 7, 8] and Auto Regressive Moving Average (ARMA) [9] are commonly used methods. All these methods have pros and cons over the others. In this project, we also apply different machine learning methods, ANN, SVM, RF, Lasso Regression (LG)[11] and Ridge Regression (RR) [10] to forecast the carbon price over time, and give an explanation for future price movements. Then, we compare those five models by analyzing model validation methods. Finally, we choose the best model for further experiments.

We have four data types: daily carbon price (CP), electricity price (EP), natural gas price (NG) and coal price (COP) that cover the period of 2009 and 2017. Prices are provided in different currencies. First of all, we work on the data to have all prices in the same currency. We completely eliminate null data. Then, graphically we investigate overall trend by smoothing the data. For analyzing data, we look for daily, monthly, yearly and seasonally time scales. For every weekday or weekends in train data set we keep a day in test data set so

that we can keep the time effect in our model. After the data management process, we apply different forecasting methods to explain future carbon price tendencies.

Key Words: Carbon Price Forecasting, Artificial Neural Network, Random Forest, Lasso Regression, Ridge Regression, Support Vector Machine

ÖZET

KARBON FİYAT ÖNGÖRME

Nurhak Karakaya

Prof. Semra Ağralı

AĞUSTOS, 2018, 88 sayfa

Son yirmi yılda dünya ekonomisi çok hızlı gelişti. Hızlı teknolojik ilerleme ve ülke ekonomilerinin hızlı bir şekilde artışı kullanılan ve ihtiyaç duyulan enerji miktarını arttırdı. Bütün ülkeler ve şirketler geleceklerini görmek için muhasebe yapmak zorundadırlar. Enerji için ayrılan bütçe ülkelerin ve şirketlerin yaptığı muhasebenin büyük bir kısmını tutmaktadır. Enerji kaynak fiyatlarındaki küçük bir artış şirketleri ve ülkeleri ciddi bir şekilde etkileyebilir. Karbon, enerjiye girdi olarak kullanılan maddelerden biridir ve karbon fiyatındaki artış bütçeleri etkileyebilir. Değişik enerji kaynaklarındaki fiyat değişimlerini tahmin etmek şirketlerin bütçesini sağlıklı bir şekilde yapmalarına yardımcı olur. Enerji fiyatları dışında, karbon kullanımı küresel ısınmayı etkileyen faktörlerden biridir.

Karbon fiyat öngörüsü için çeşitli makine öğrenimi metotları kullanılmaktadır. Değişik akademik makaleleri incelediğimizde en çok kullanılan metotlar: Destek Vektör Makinası (SVM) [1],[2],[3], Rassal Orman (RF)[4],[5] ve Yapay Sinir Ağları (ANN)[6],[7],[8] dır. Bunlara ek olarak Autoregressif Kayan Ortalama (ARMA)[9] da oldukça sık kullanılan yöntemlerden biridir. Bu tekniklerin diğerlerine göre avantaj ve dezavantajları vardır. Bazı veri setlerinde bu yöntemlerden biri çok iyi sonuç verirken bazılarında beklenen sonuçları vermemektedir. Bu projede de karbon fiyat öngörüsü için ANN, RF, SVM, Ridge Regresyonu (RR) [10] ve Lasso Regresyonu (LR)[11] kullanılacaktır. Çalışma boyunca oluşturulacak test ve eğitim veri setleri ile değişik metotlar ile en iyi modeli bulmaya çalışacağız. Daha sonra değişik model tasdik yöntemleri ile en iyi model seçilecektir.

Çalışmada kullanacağımız veri kümelerinde ilgili gün ve o güne ait karbon, elektrik, doğal gaz ve kömür fiyatı bulunmaktadır. Veriler 2009 - 2017 yılları arasını kapsamaktadır. İlk iş olarak bütün verileri ortak bir para biriminde tutmamız gerekmektedir. Buna ek olarak iyi bir tahminlime yapabilmek için zamana göre çalışma yapılmalıdır. Günlük, Haftalık, aylık, dönemsel ve yıllık çalışmalar yapılacaktır. Hafta içi ve hafta sonu fiyat değişimlerini ölçmek

için test ve eğitim veri setleri bu iki gün tipinde tutacak şekilde ayarlanacaktır. Sonrasında modeller çalıştırılarak en iyi sonuç veren model seçilecektir.

Anahtar Kelimeler: Karbon Fiyat Öngörme, Yapay Sinir Ağları, Rassal Orman, Makina Öğrenme, Lasso Regresyonu, Ridge Regresyonu, Destek Vektör Makinası, Agresif Kayan Ortalama

TABLE OF CONTENTS

1. INTRODUCTION

In this project we forecast carbon prices by using carbon prices and some other prices that are traded under European Union's Emissions Trading System (EU ETS). In the project, we have four main sources: carbon, coal, natural gas and electricity. These are the overall data source for our project. The prices of these four items are in different currencies. Our first task in the project is to convert all of them into USD Dollar. During the project, we call “Carbon price”, “Coal price”, ”Natural gas price” and “Electricity price” as our main attributes.

We have prices that cover 7 years. In section 3, we combine all of those prices into a single data table. In section 3, we create new numerical attributes from our main attributes. Moreover; in section 3, we create new attributes from Date attribute. For forecasting; we use SVM, RF, RR, LR and ANN. Before forecasting in section 4, we use forward [16] and backward [17] selection algorithms to find most predictive attributes. Then by using most predictive attributes, we use machine learning methods to forecast carbon price. All of the models in our project use the same attributes. And all of the models use same kind of data except for ANN after log transformation we use min max scaler for ANN. In our project, we try to understand how forecasting is affected by the amount of data. So, we create two scenarios for forecasting. In scenario 1, we have three years for training and one year for testing. In scenario 2, we have five years for training and 1 year for testing. After that we run five machine learning methods for those two scenarios.

In section 2, we define project methodology and what is in our project’s scope. In section 5, we compare all of the model results. In section 5, we also compare the results of two scenarios.

2. PROJECT DEFINITION

In this project, we analyze the carbon prices that are traded under European Union's Emissions Trading System (EU ETS) [13]. To investigate carbon prices, we use coal, natural gas and electricity prices. We firstly discover overall data and then use five different methods to investigate carbon prices. We do not take into account economic crises or inflations in USA in that period. The decline in the value of USD is not in the scope of this project. The currency exchanges are done in its own day.

This project aims to develop a prediction model that anticipates future carbon prices given a real-world data set. In this project, we used SVM, RR, RF, ANN and LR machine learning methods. We will compare the models with respect to Root Mean Square Error (RMSE). The overall processes for this project are given in Figure 1.

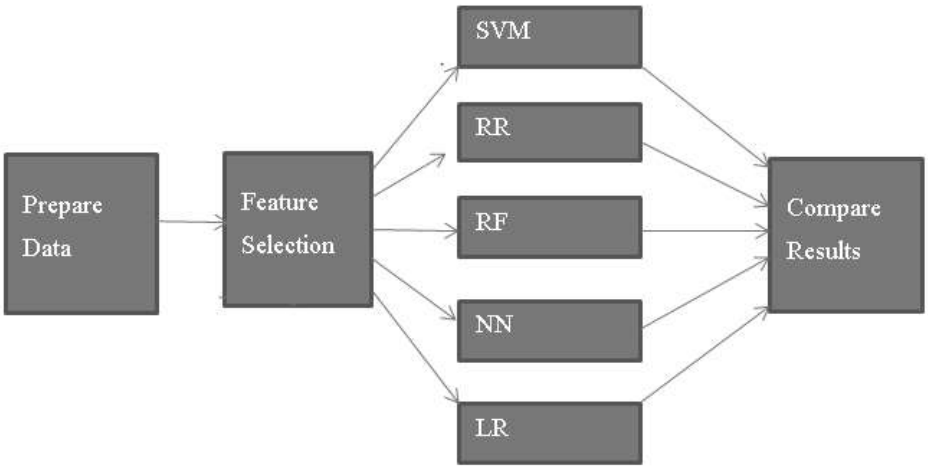

Figure 1. Overall project structure

For attributes in this project, we use attributes, features and predictors interchangeably.

We divide the overall Project into three parts: Data discovery, modeling and comparing the results. The first part is the data discovery part. In that part, we apply data preparation and analysis steps. For data discovery, we work on Python 3.7. For modeling and comparing the result part, we work on R.

3. DATA DISCOVERY

The data in the project are provided in different MS Excel files. There are 4 files for: carbon, coal, natural gas and electricity prices. Moreover, there are four files that show different currencies. The prices are provided in different currencies (Coal in USD, Carbon in Euro, Natural Gas and Electricity in GBP), so we have to convert all those currencies into a single currency. We choose United States Dollars (USD) since it is commonly used in Turkey. All of the files have two columns one indicating the date and second indicating the price of the source on that date (Table 1).

Table 1. First nine rows of a sample data file

Date	PX_LAST
02.01.2013	6,52
03.01.2013	6,22
04.01.2013	6,21
07.01.2013	6,49
08.01.2013	6,29
09.01.2013	6,05
10.01.2013	5,89
11.01.2013	5,73
14.01.2013	5,7

We load data files into different data frames. The prices are in different currencies. To work on it, we convert all currencies into USD. Then, we join all data frames into one unique data frame. After that, all of the operations are implemented on that unique data frame.

The data statistics for those 4 prices are given in Table 2. Our target column in that project is the “CarbonPriceInUSD” column. Its price is the lowest of all prices. The distributions of all prices look like normal.

Table 2. Statistical values for main four attributes

CarbonPriceInUSD	CoalPriceInUSD	ElectricityPriceInUSD	NaturalGasPriceInUSD
Min. : 3.493	Min. : 42.00	Min. : 43.57	Min. : 30.13
1st Qu.: 6.589	1st Qu.: 68.80	1st Qu.: 58.34	1st Qu.: 53.18
Median : 8.666	Median : 78.38	Median : 65.71	Median : 70.08
Mean :11.143	Mean : 81.01	Mean : 66.82	Mean : 73.25
3rd Qu.:17.563	3rd Qu.: 91.24	3rd Qu.: 75.57	3rd Qu.: 92.95
Max. :24.806	Max. :132.60	Max. :102.55	Max. :120.56

3.1. Data Visualization

The data files start from 2009 and end with 2017. The prices change over time. In Figure 2, we show the overall trend of carbon prices between 2009 and 2017. The Figure is created by R ggplot package [12]. The light blue is the actual values of carbon price. The other three lines are created by Smoothed Conditional Means (SCM) [18]. For SCM, we use goem_smooth function of ggplot package. The red line uses method “auto”, the green line uses method “loess” and the dark blue lines uses method “lm” . In Figure 2, we set Y axis [0-30] because carbon prices are between 3 and 24 (Table 2). From Figure 2, we see that the prices initially increase a bit then there is a sharp decrease in prices and then it turns into more steady condition.

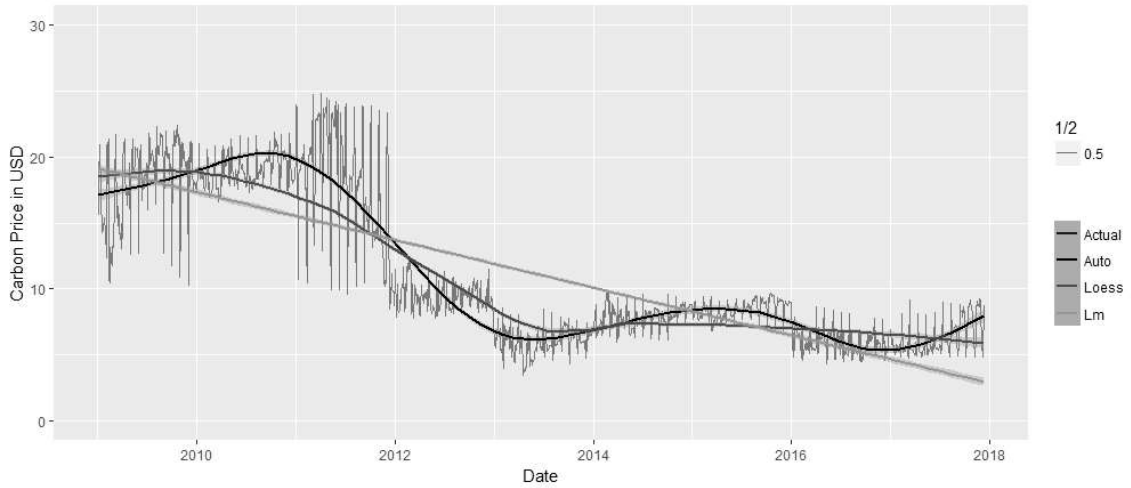

Figure 2: Carbon prices from January 2009 to December 2017

In Figure 3, we see the life cycle of coal price. Again the time interval for coal price is same as time interval for carbon price. For coal price, we use again same packages. And again here: the red line uses method “auto”, the green line uses method “loess” and the dark blue lines uses method “lm” . In Figure 3, we set Y axis in range of [0-125] (Table 2). The coal price life

cycle looks like carbon price life cycle. The price increases about for a three years. Then starts to decrease for a long time about four years and then again increases.

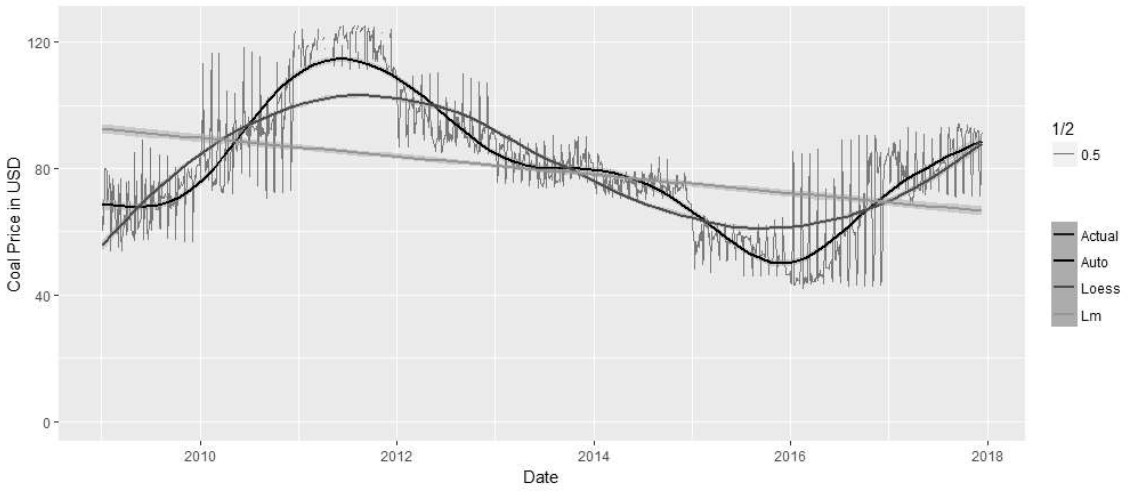

Figure 3: Coal prices from January 2009 to December 2017

In Figure 4 and Figure 5, we see the price changes in natural gas and electricity. In Figure 4 and Figure 5, we set Y axis in range of [0-125] (Table 2). The prices for natural gas again more strictly increases and decreases. The prices for electricity are more stationary. And here also we use same methods (red line “auto”, green line “loess” , dark blue lines “lm”) .

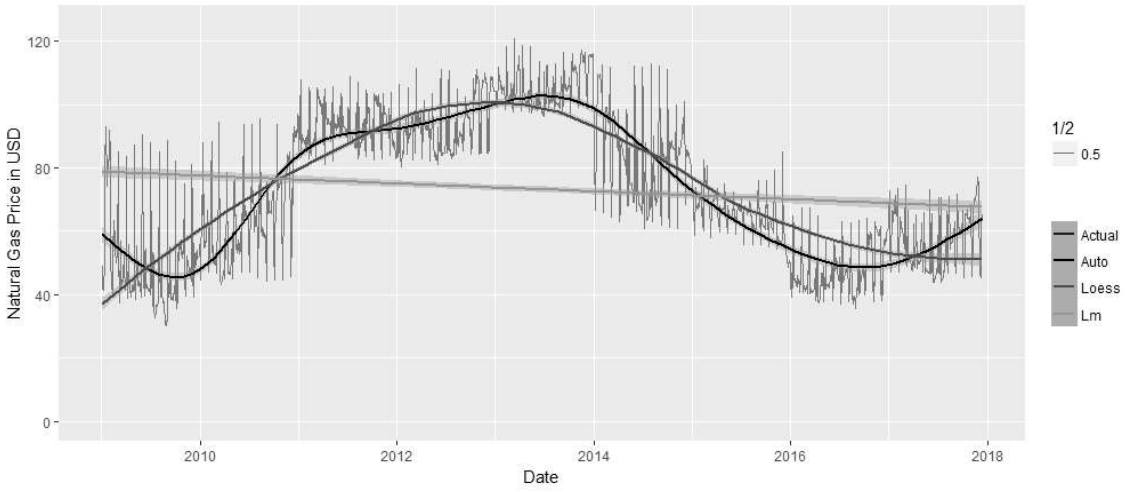

Figure 4: Natural gas prices from January 2009 to December 2017

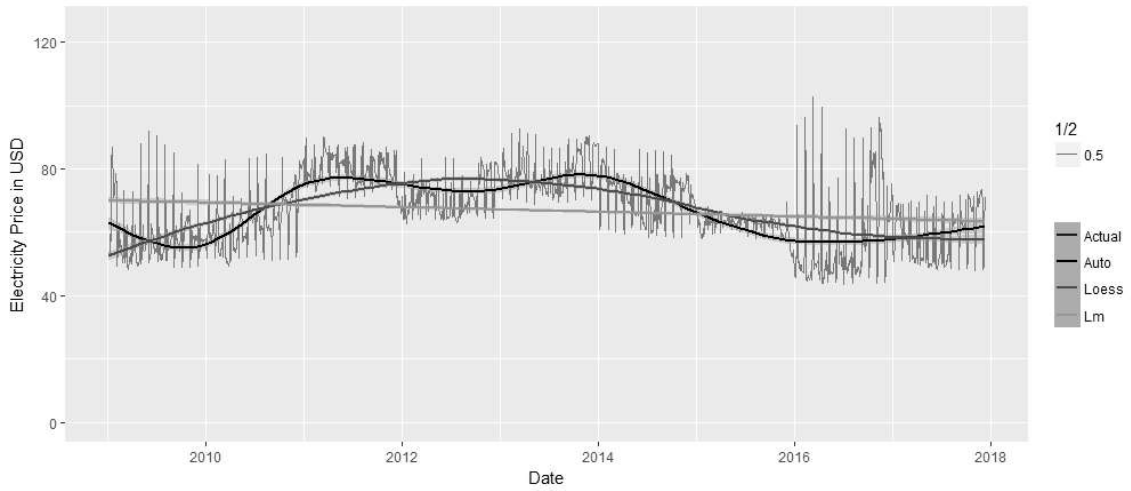

Figure 5: Electricity prices from January 2009 to December 2017

In Figure 6, we combine all Figures (figure 2-5) to see the relation between four prices in our dataset; carbon. coal, natural gas and electricity. Here we use geom_smooth function with method “auto”.

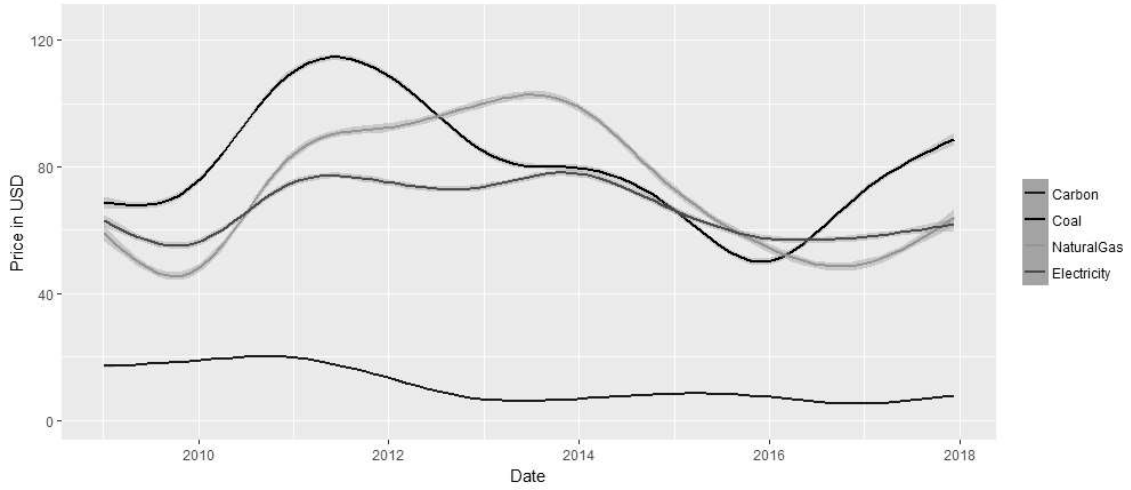

Figure 6: SCM auto prices for Carbon, Coal, Natural Gas and Electricity from January 2009 to December 2017.

3.2. New Features

Mei et al. [5] created four attributes to keep prices of previous hours and 3 attributes for time definitions. For new attributes, we have four steps. In step one; we create new attributes by using our main attributes in Table 2. In step two; we create new attributes by using Date attribute. In step 3, we create dummy variables from categorical variables that we create in step 2 and delete original categorical variables. And finally in step four, we take log transformation for numerical variables that we create in step 1.

3.2.1. Features from Main Features

The names of our four main features are: “CarbonPriceInUSD”, “CoalPriceInUSD”, “ElectricityPriceInUSD” and NaturalGasPriceInUSD. For every attribute in main attributes, we create the following new attributes in Table 3, in Table 4, in Table 5 and in Table 6.

Table 3. New attributes created by “CarbonPriceInUSD”

Predictor Name	Description
CarbonPriceInUSD_D1	Price of Carbon one day ago
CarbonPriceInUSD _D2	Price of Carbon two days ago
CarbonPriceInUSD _D3	Price of Carbon three days ago
CarbonPriceInUSD _C12	[Price of Carbon one day ago] - [Price of Carbon two days ago]
CarbonPriceInUSD _CR12	[Price of Carbon one day ago] / [Price of Carbon two days ago]
CarbonPriceInUSD _C23	[Price of Carbon two days ago] - [Price of Carbon three days ago]
CarbonPriceInUSD _CR23	[Price of Carbon two days ago] / [Price of Carbon three days ago]
CarbonPriceInUSD _W1	Price of Carbon one week ago
CarbonPriceInUSD _AW1	Average Price of Carbon for last week (last 7 days)
CarbonPriceInUSD _W2	Price of Carbon two weeks ago
CarbonPriceInUSD _W3	Price of Carbon three weeks ago
CarbonPriceInUSD _M1	Price of Carbon one month ago (30 days ago)
CarbonPriceInUSD _AM1	Average Price of Carbon for last month (last 30 days)
CarbonPriceInUSD _M2	Price of Carbon two months ago (60 days ago)
CarbonPriceInUSD _M3	Price of Carbon three months ago (90 days ago)
CarbonPriceInUSD _AM3	Average Price of Carbon for last three month (last 90 days)

Table 4. New attributes created by “CoalPriceInUSD”

Predictor Name	Description
CoalPriceInUSD_D1	Price of Coal one day ago
CoalPriceInUSD _D2	Price of Coal two days ago
CoalPriceInUSD _D3	Price of Coal three days ago
CoalPriceInUSD _C12	[Price of Coal one day ago] - [Price of Coal two days ago]
CoalPriceInUSD _CR12	[Price of Coal one day ago] / [Price of Coal two days ago]
CoalPriceInUSD _C23	[Price of Coal two days ago] - [Price of Coal three days ago]
CoalPriceInUSD _CR23	[Price of Coal two days ago] / [Price of Coal three days ago]
CoalPriceInUSD _W1	Price of Coal one week ago
CoalPriceInUSD _AW1	Average Price of Coal for last week (last 7 days)
CoalPriceInUSD _W2	Price of Coal two weeks ago
CoalPriceInUSD _W3	Price of Coal three weeks ago
CoalPriceInUSD _M1	Price of Coal one month ago (30 days ago)
CoalPriceInUSD _AM1	Average Price of Coal for last month (last 30 days)
CoalPriceInUSD _M2	Price of Coal two months ago (60 days ago)
CoalPriceInUSD _M3	Price of Coal three months ago (90 days ago)
CoalPriceInUSD _AM3	Average Price of Coal for last three month (last 90 days)

Table 5. New attributes created by “ElectricityInUSD”

Predictor Name	Description
ElectricityPriceInUSD_D1	Price of Electricity one day ago
ElectricityPriceInUSD _D2	Price of Electricity two days ago
ElectricityPriceInUSD _D3	Price of Electricity three days ago
ElectricityPriceInUSD _C12	[Price of Electricity one day ago] - [Price of Electricity two days ago]
ElectricityPriceInUSD _CR12	[Price of Electricity one day ago] / [Price of Electricity two days ago]
ElectricityPriceInUSD _C23	[Price of Electricity two days ago] - [Price of Electricity three days ago]
ElectricityPriceInUSD _CR23	[Price of Electricity two days ago] / [Price of Electricity three days ago]
ElectricityPriceInUSD _W1	Price of Electricity one week ago
ElectricityPriceInUSD _AW1	Average Price of Electricity for last week (last 7 days)
ElectricityPriceInUSD _W2	Price of Electricity two weeks ago
ElectricityPriceInUSD _W3	Price of Electricity three weeks ago
ElectricityPriceInUSD _M1	Price of Electricity one month ago (30 days ago)
ElectricityPriceInUSD _AM1	Average Price of Electricity for last month (last 30 days)
ElectricityPriceInUSD _M2	Price of Electricity two months ago (60 days ago)
ElectricityPriceInUSD _M3	Price of Electricity three months ago (90 days ago)
ElectricityPriceInUSD _AM3	Average Price of Electricity for last three month (last 90 days)

Table 6. New attributes created by “NaturalGasInUSD”

Predictor Name	Description
NaturalGasPriceInUSD_D1	Price of Natural Gas one day ago
NaturalGasPriceInUSD _D2	Price of Natural Gas two days ago
NaturalGasPriceInUSD _D3	Price of Natural Gas three days ago
NaturalGasPriceInUSD _C12	[Price of Natural Gas one day ago] - [Price of Natural Gas two days ago]
NaturalGasPriceInUSD _CR12	[Price of Natural Gas one day ago] / [Price of Natural Gas two days ago]
NaturalGasPriceInUSD _C23	[Price of Natural Gas two days ago] - [Price of Natural Gas three days ago]
NaturalGasPriceInUSD _CR23	[Price of Natural Gas two days ago] / [Price of Natural Gas three days ago]
NaturalGasPriceInUSD _W1	Price of Natural Gas one week ago
NaturalGasPriceInUSD _AW1	Average Price of Natural Gas for last week (last 7 days)
NaturalGasPriceInUSD _W2	Price of Natural Gas two weeks ago
NaturalGasPriceInUSD _W3	Price of Natural Gas three weeks ago
NaturalGasPriceInUSD _M1	Price of Natural Gas one month ago (30 days ago)
NaturalGasPriceInUSD _AM1	Average Price of Natural Gas for last month (last 30 days)
NaturalGasPriceInUSD _M2	Price of Natural Gas two months ago (60 days ago)
NaturalGasPriceInUSD _M3	Price of Natural Gas three months ago (90 days ago)
NaturalGasPriceInUSD _AM3	Average Price of Natural Gas for last three month (last 90 days)

3.2.2. Features from Date

From Date attribute we create new time attributes as in Table 7. The time attributes are: Year, Month, Weekday and Quarter and Season. Note: all of those attributes are categorical variables accept Year, Month is also categorical variable.

Table 7. New attributes created by Date

Predictor	Description
Month	Month of the Date. From 1 to12
Year	Year of the Date. From 2009 to 2017
Weekday	Weekday of the Date. From Sun (Sunday)to Sat (Saturday).
Quarter	İf Month in (1,2,3) Then First else if Month in (4,5,6) Then Second else if Month in (7,8,9) Then Third else Fourth
Season	If Date between 21-March to 20-June Then Spring Else If Date between 21-June to 22-September then Summer Else if Date between 23-September to 20-December Then Autumn Else Winter

3.2.3. Dummy Features from Categorical Features

In this project, we create regression models so categorical features should be transformed into dummy features [15]. In section 3.2.2 we create four categorical variables (Month, Weekday, Quarter and Season).

- Month has the values from 1 to 12.
- Weekday has the values of (“Mon”, ”Tue”, ”Wed”, ”Thu”, ”Fri”, ”Sat”, ”Sun”).
- Quarter has the values of (“First”, ”Second”, “Third”, ”Fourth”).
- Season has the values of (“Spring”, “Summer”, ”Autumn”, “Winter”)

For every category in a categorical feature, we create [number of category – 1] new features. For example, we create “Weekday_Mon”, “Weekday_Tue”, “Weekday_Wed”, “Weekday_Thu”, “Weekday_Sat”, “Weekday_Sun” for Weekday variable. If the day is Monday for example, then for “Weekday_Mon” has a value of 1 and others has value of 0. All of the days have the same rules. And for Friday we don’t create a variable. For Friday every dummy variable has value of zero.

Table 8. Dummy variables for “Weekday”

Weekday Values	Weekday_Mon	Weekday_Tue	Weekday_Wed	Weekday_Thu	Weekday_Sat	Weekday_Sun
Monday	1	0	0	0	0	0
Tuesday	0	1	0	0	0	0
Wednesday	0	0	1	0	0	0
Thursday	0	0	0	1	0	0
Friday	0	0	0	0	0	0
Saturday	0	0	0	0	1	0
Sunday	0	0	0	0	0	1

For Season, Month and Quarter we create dummy variables with same rule. Month has 11 dummy variables. Quarter and Season has 3 dummy variables. After creating dummy variables we delete original categorical variables.

3.2.4. Log Transformation

Keles et al. [14] normalized data then put normalized attributes to ANN. We take log transformation of numeric variables and then used those variables for our models. For normalization we took R “skewness” package [19] to find volatile attribute and then we use R log function [20] to take logarithm of numeric attributes. Now in our data set we have: Date feature, log transformed numeric features and dummy features. In total we have 93 features. All of the features in our project are in table 9.

Table 9. All features in our data set

[1] "Date"	"CarbonPriceInUSD"	"CoalPriceInUSD"	"ElectricityPriceInUSD"
[5] "NaturalGasPriceInUSD"	"CarbonPriceInUSD_D1"	"CarbonPriceInUSD_D2"	"CarbonPriceInUSD_D3"
[9] "CarbonPriceInUSD_C12"	"CarbonPriceInUSD_CR12"	"CarbonPriceInUSD_C23"	"CarbonPriceInUSD_CR23"
[13] "CarbonPriceInUSD_W1"	"CarbonPriceInUSD_AW1"	"CarbonPriceInUSD_W2"	"CarbonPriceInUSD_W3"
[17] "CarbonPriceInUSD_M1"	"CarbonPriceInUSD_AM1"	"CarbonPriceInUSD_M2"	"CarbonPriceInUSD_M3"
[21] "CarbonPriceInUSD_AM3"	"CoalPriceInUSD_D1"	"CoalPriceInUSD_D2"	"CoalPriceInUSD_D3"
[25] "CoalPriceInUSD_C12"	"CoalPriceInUSD_CR12"	"CoalPriceInUSD_C23"	"CoalPriceInUSD_CR23"
[29] "CoalPriceInUSD_W1"	"CoalPriceInUSD_AW1"	"CoalPriceInUSD_W2"	"CoalPriceInUSD_W3"
[33] "CoalPriceInUSD_M1"	"CoalPriceInUSD_AM1"	"CoalPriceInUSD_M2"	"CoalPriceInUSD_M3"
[37] "CoalPriceInUSD_AM3"	"ElectricityPriceInUSD_D1"	"ElectricityPriceInUSD_D2"	"ElectricityPriceInUSD_D3"
[41] "ElectricityPriceInUSD_C12"	"ElectricityPriceInUSD_CR12"	"ElectricityPriceInUSD_C23"	"ElectricityPriceInUSD_CR23"
[45] "ElectricityPriceInUSD_W1"	"ElectricityPriceInUSD_AW1"	"ElectricityPriceInUSD_W2"	"ElectricityPriceInUSD_W3"
[49] "ElectricityPriceInUSD_M1"	"ElectricityPriceInUSD_AM1"	"ElectricityPriceInUSD_M2"	"ElectricityPriceInUSD_M3"
[53] "ElectricityPriceInUSD_AM3"	"NaturalGasPriceInUSD_D1"	"NaturalGasPriceInUSD_D2"	"NaturalGasPriceInUSD_D3"
[57] "NaturalGasPriceInUSD_C12"	"NaturalGasPriceInUSD_CR12"	"NaturalGasPriceInUSD_C23"	"NaturalGasPriceInUSD_CR23"
[61] "NaturalGasPriceInUSD_W1"	"NaturalGasPriceInUSD_AW1"	"NaturalGasPriceInUSD_W2"	"NaturalGasPriceInUSD_W3"
[65] "NaturalGasPriceInUSD_M1"	"NaturalGasPriceInUSD_AM1"	"NaturalGasPriceInUSD_M2"	"NaturalGasPriceInUSD_M3"
[69] "NaturalGasPriceInUSD_AM3"	"Year"	"Month_10"	"Month_11"
[73] "Month_12"	"Month_2"	"Month_3"	"Month_4"
[77] "Month_5"	"Month_6"	"Month_7"	"Month_8"
[81] "Month_9"	"Weekday_Mon"	"Weekday_Sat"	"Weekday_Sun"
[85] "Weekday_Thu"	"Weekday_Tue"	"Weekday_Wed"	"Quarter_Fourth"
[89] "Quarter_Second"	"Quarter_Third"	"Season_Spring"	"Season_Summer"
[93] "Season_Winter"			

4. MODELING

In our project, we use two types of data set. In our first data set; we use Year in (2010, 2011, 2012) for training and 2013 for testing, 2013 is not included in training data set. In our second data set, we use Year in (2010, 2011, 2012, 2013, 2014) in our training set and 2015 in our test set, 2013 included in training set. From Figure 2, we see in carbon prices there is a sharp decrease from 2011 to 2013 and 2013 has the lowest prices.

This project is a regression problem. We try 5 different machine learning regression methodologies: Ridge and Lasso Regression, SVM, RF and ANN. We use all of algorithms for two data sets.

Before modeling, we use feature selection techniques to find optimal features that can be used for our models. We use forward [16] and backward [17] feature selection techniques to find optimal features. We choose number of features with lowest RMSE. We will calculate lowest RMSE for forward selection and lowest RMSE for backward selection and then choose the attributes with the lowest overall RMSE.

4.1. First Scenario (Training Set: 2010-2012, Test Set: 2013)

4.1.1. Feature Selection for First Scenario

We have 93 features in our data set, 92 predictors and 1 target. In order to decrease number of features we try two different methods:

First: We calculate the correlation of attributes with the target attribute (CarbonPriceInUSD). Some attributes have NA correlation with target attribute. We discard those attributes from our data set which are highly correlated with other attributes and can cause over fitting. The attributes in Table 10 are eliminated. 4 predictors are eliminated and 88 predictors are still candidate.

Table 10.Eliminated attributes because of correlation with target

```
#[1] "CarbonPriceInUSD_C12" "CarbonPriceInUSD_C23" "CoalPriceInUSD_C12"   "CoalPriceInUSD_C23"
```

After that we use forward [16] and backward [17] feature selection techniques to find most important attributes to forecast carbon price. We use 10-fold forward [21] and 10-fold backward [21] cross validation to find best predictive attributes.

We firstly calculate The MSE for forward selection. In Table 11, we see the number of attribute and their RMSE results. We should choose the attribute with best RMSE, which is the lowest RMSE. 12 attributes give the best result for forward selection. The minimum RMSE is 0.007239524839, which is at 12 attributes.

Table 11. RMSEs for forward selection and number of attributes

1	2	3	4	5	6	7	8	9	10	11
0.011918090	0.011100052	0.011269746	0.010674545	0.008849755	0.008077233	0.007725398	0.007601242	0.007459614	0.007367004	0.007280341
12	13	14	15	16	17	18	19	20	21	22
0.007239525	0.007389452	0.007331865	0.007382291	0.007443778	0.007427134	0.007438706	0.007363925	0.007358821	0.007392169	0.007334773
23	24	25	26	27	28	29	30	31	32	33
0.007351041	0.007358797	0.007316722	0.007339892	0.007373029	0.007371908	0.007378612	0.007372606	0.007376789	0.007335648	0.007295154
34	35	36	37	38	39	40	41	42	43	44
0.007304491	0.007304500	0.007270379	0.007272925	0.007294734	0.007279015	0.007270180	0.007348426	0.007365685	0.007354552	0.007350442
45	46	47	48	49	50	51	52	53	54	55
0.007338185	0.007344638	0.007403529	0.007385936	0.007358819	0.007343331	0.007393277	0.007401918	0.007424140	0.007425523	0.007452274
56	57	58	59	60	61	62	63	64	65	66
0.007500329	0.007538409	0.007525270	0.007526768	0.007493398	0.007549975	0.007551164	0.007561791	0.007545791	0.007540782	0.007569845
67	68	69	70	71	72	73	74	75	76	77
0.007530677	0.007506653	0.007526447	0.007459066	0.007491376	0.007443984	0.007450726	0.007456007	0.007466460	0.007450449	0.007454394
78	79	80	81	82	83	84	85	86	87	88
0.007412225	0.007402855	0.007438675	0.007433780	NA	NA	NA	NA	NA	NA	NA

From Figure 7, we see that when the number of attributes increases initially RMSE decreases sharply after that it gets more stationary.

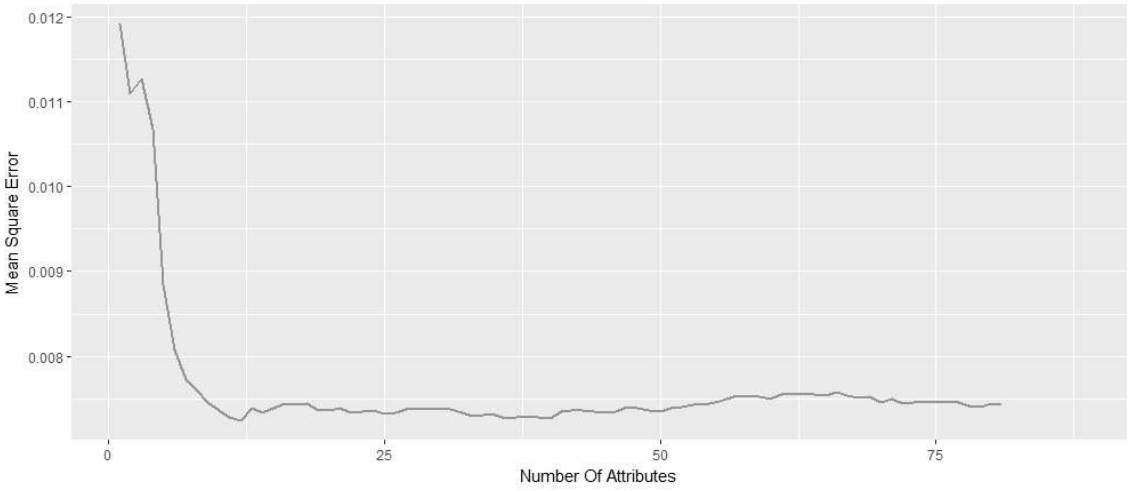

Figure 7: MSE for forward selection and number of attributes

In Table 12, we list the best predictive useful features (12 features) for forward selection.

Table 12. Predictive features for forward selection

[1] "ElectricityPriceInUSD"	"NaturalGasPriceInUSD"	"CarbonPriceInUSD_D1"	"CarbonPriceInUSD_W3"
[5] "CarbonPriceInUSD_AM1"	"CarbonPriceInUSD_M2"	"ElectricityPriceInUSD_D1"	"NaturalGasPriceInUSD_D3"
[9] "NaturalGasPriceInUSD_M3"	"Month_10"	"Month_6"	"ElectricityPriceInUSD_C23"

In Table 13, we see the number of attribute and RMSE results for backward selection. 52 attributes give the best result for backward selection. The minimum of RMSE is 0.007242504 which is slightly higher than forward best RMSE 0.007239524839.

Table 13. RMSEs for backward selection and number of attributes

1	2	3	4	5	6	7	8	9	10	11
0.054750173	0.013835392	0.008106638	0.007585324	0.007516004	0.007489595	0.007516976	0.007715472	0.007620187	0.007562363	0.007616630
12	13	14	15	16	17	18	19	20	21	22
0.007563206	0.007592903	0.007591524	0.007562556	0.007623293	0.007608923	0.007545732	0.007537093	0.007506745	0.007440327	0.007429712
23	24	25	26	27	28	29	30	31	32	33
0.007440316	0.007435550	0.007452586	0.007501014	0.007490651	0.007454179	0.007477386	0.007423206	0.007419403	0.007442958	0.007414106
34	35	36	37	38	39	40	41	42	43	44
0.007465949	0.007511728	0.007630073	0.007586394	0.007593356	0.007595152	0.007551992	0.007475506	0.007456203	0.007431114	0.007431904
45	46	47	48	49	50	51	52	53	54	55
0.007342660	0.007339432	0.007319064	0.007306823	0.007272592	0.007265587	0.007252574	0.007242504	0.007256363	0.007296841	0.007250367
56	57	58	59	60	61	62	63	64	65	66
0.007296751	0.007316080	0.007333795	0.007355494	0.007356505	0.007404435	0.007429819	0.007254593	0.007264267	0.007296775	0.007328318
67	68	69	70	71	72	73	74	75	76	77
0.007347544	0.007369184	0.007367420	0.007364028	0.007425878	0.007409117	0.007381483	0.007415183	0.007493049	0.007509053	0.007509116
78	79	80	81	82	83	84	85	86	87	88
0.007500456	0.007527251	0.007532020	0.007527092	NA	NA	NA	NA	NA	NA	NA

From Figure 8, we see that when the number of attributes increases initially RMSE decreases sharply after that it gets more stationary.

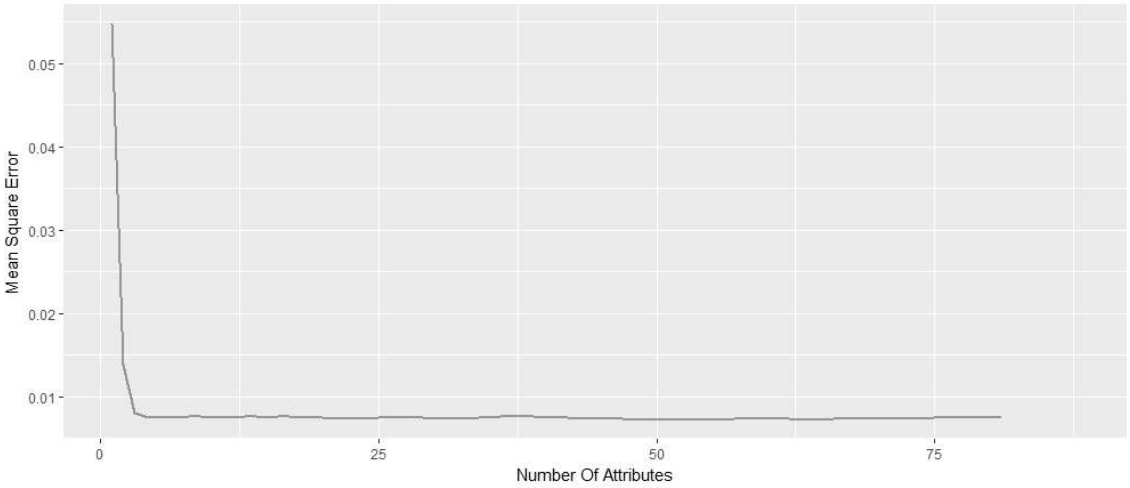

Figure 8: RMSE for backward selection and number of attributes

In Table 14, we list the best useful features (52 features) for backward selection.

Table 14. Predictive features for backward selection

[1]	"Date"	"CoalPriceInUSD"	"ElectricityPriceInUSD"	"NaturalGasPriceInUSD"
[5]	"CarbonPriceInUSD_D1"	"CarbonPriceInUSD_D2"	"CarbonPriceInUSD_D3"	"CarbonPriceInUSD_CR12"
[9]	"CarbonPriceInUSD_CR23"	"CarbonPriceInUSD_W1"	"CarbonPriceInUSD_W2"	"CarbonPriceInUSD_W3"
[13]	"CarbonPriceInUSD_M2"	"CarbonPriceInUSD_M3"	"CarbonPriceInUSD_AM3"	"CoalPriceInUSD_D1"
[17]	"CoalPriceInUSD_D2"	"CoalPriceInUSD_D3"	"CoalPriceInUSD_CR12"	"CoalPriceInUSD_W1"
[21]	"CoalPriceInUSD_AW1"	"CoalPriceInUSD_W2"	"CoalPriceInUSD_AM1"	"CoalPriceInUSD_M2"
[25]	"CoalPriceInUSD_AM3"	"ElectricityPriceInUSD_D1"	"ElectricityPriceInUSD_D2"	"ElectricityPriceInUSD_D3"
[29]	"ElectricityPriceInUSD_AW1"	"ElectricityPriceInUSD_W2"	"ElectricityPriceInUSD_W3"	"ElectricityPriceInUSD_AM1"
[33]	"ElectricityPriceInUSD_AM3"	"NaturalGasPriceInUSD_D2"	"NaturalGasPriceInUSD_D3"	"NaturalGasPriceInUSD_CR23"
[37]	"NaturalGasPriceInUSD_AW1"	"NaturalGasPriceInUSD_W3"	"NaturalGasPriceInUSD_M3"	"NaturalGasPriceInUSD_AM3"
[41]	"Month_11"	"Month_12"	"Month_2"	"Month_3"
[45]	"Month_6"	"Month_7"	"Month_9"	"Weekday_Sun"
[49]	"Weekday_Thu"	"Quarter_Fourth"	"Quarter_Second"	"Quarter_Third"

Since the RMSE for forward selection is less than the RMSE for backward selection, we use the important attributes that come from forward selection. The selected attributes are in Table 12. For our models we use those predictors.

4.1.2. Ridge Regression (RR) for First Scenario

RR [10] is used for regression problems. It normally looks like Residual Sum of Squares (RSS) except that it has a shrinkage penalty. When a predictor is not powerful to explain the target attribute, RR gives a penalty to that predictor so that that predictor becomes

less affective for describing the model. In that case, the model does not add value from that predictor. We call this as "shrinkage penalty" [25]. The parameter in RR that defines the penalty is called lambda or tuning parameter.

We use "glmnet" package [22] for RR. When the alpha parameter = 0, glmnet behaves as RR when alpha = 1 glmlet behaves as LR. In RR there is just one parameter to optimize. It is alpha parameter. Glmlet has its own grid search [23] algorithm. We used that grid search algorithm with 10-fold cross validation to find best lambda value for our data set. The best lambda value for RR is 0.03373075301; we used that value in prediction part. For model creation part, we used lambda as a sequence from 10^{10} to 10^{-2}.

When we run test set with the tuned RR model that we find from grid search. RMSE for RR is 0.01698475748. In Figure 9, the green line is the actual values the red line is the predicted results. And the blue line is the difference between predicted – actual.

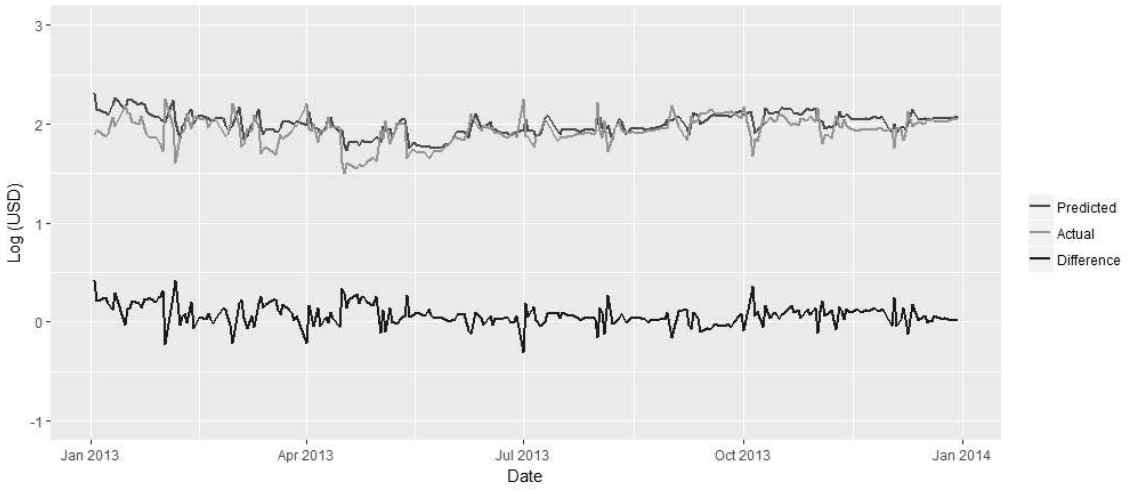

Figure 9: Predicted and Actual results for RR

4.1.3. Lasso Regression for First Scenario

RR will include all p predictors in the final model. This may not be problem for model prediction but can lead to which p predictors are important. Increasing the value of lambda

will reduce parameter importance but never exclude any parameter from the model. To overcome this, LR [11] replaces RR penalty with LR penalty.

LR = RSS + lambda * *[j=1..p Sum of $|b_j|$], where p is the final predictors.

We took absolute value. And when an item is useless LR discards it from predictor list.

As in the RR, we use "glmnet" package for LR. Now the alpha parameter = 1. We use grid search algorism of glmnet with 10-fold cross validation to find best lambda value for our data set. The best lambda value for LR is 0.0001975617223, we use that value in prediction part. For model creation part, we used lambda as a sequence from 10^{10} to 10^{-2}.

When we run test set with the tuned RR model that we find from grid search. RMSE for LR is 0.01350746101which is better than the RMSE for RR 0.01698475748. In Figure 10 we can see the LR results. The green line is the actual values the red line is the predicted results. And the blue line is the difference between predicted – actual.

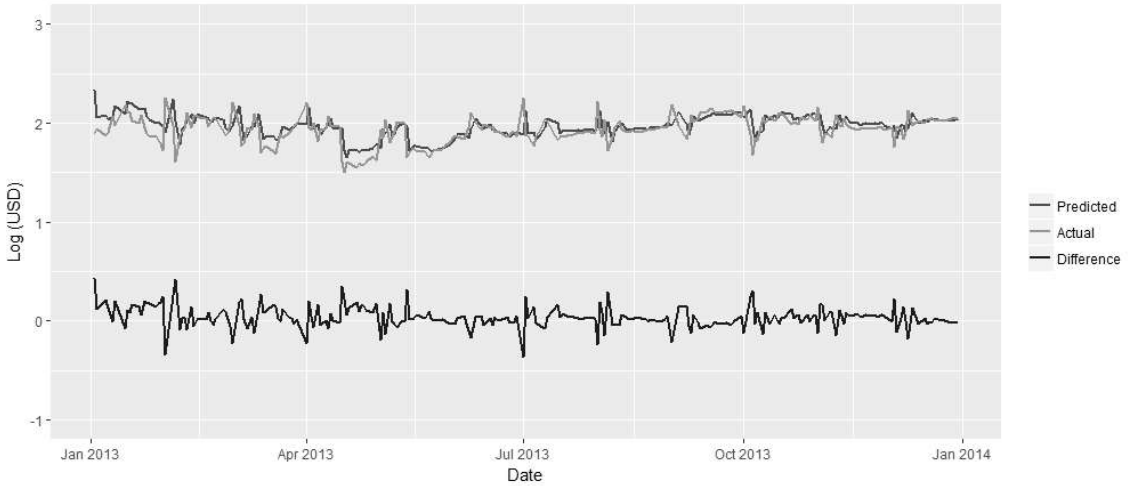

Figure 10: Predicted and Actual results for LR

4.1.4. Random Forest (RF) for First Scenario

In RF there are two important parameters that should be optimized. The number of tree and the number of candidate predictors.

For RF we use our own grid search algorithm with 10-fold CV. In our algorithm, we use three for loops one inside another. For first loop, we take mtree (number of tree) as c(50,100,500). For second loop, we take mtry (number of candidate predictors) as c(3,5,10). The third loop is for 10-fold CV. In every time in 10-fold CV, we take one tenth of data for test case and the remaining for training set.

For mt in mtree loop:
    For tr in mtry loop:
        For i in 1 to 10 loop:
            Get MSE
        End loop
        Get average of MSE
    End Loop
End Loop

From that loop, we calculate the RMSE for every case. Every mtree* mtry combination run 10 times and we calculate the average of RMSE for every mtree* mtry combination. The result for average RMSE is at Table 15:

Table 15. RMSE results for first RF for grid search

mtry	3	5	10
mtree			
# 50	0.01048196690	0.009400864890	0.008974656508
# 100	0.01038480785	0.009139922028	0.009007893467
# 500	0.01039172742	0.009389295902	0.009316891856

The minimum MSE is 0.008974656508. That means mtree=50 and mtry=10 gives the best result.

Then we take another grid search with 10-fold CV. In this grid search we use parameter values that are close to previous best result. In our second run; we take ntree as c(40,50,60), which are close to 50, and mtry as c(9,10,11)), which are close to 10. Our new

results are in Table 16. The best result has RMSE 0.008940246528. That means mtree=40 and mtry=10 gives the best result. Since "ntree=40 and mtry=10" combination gives better RMSE, we take ntree=40 and mtry=10 for RF parameters.

Table 16. RMSE results for second RF for grid search

mtry	9	10	11
mtree			
# 40	0.009796510783	0.008940246528	0.009397389415
# 50	0.009812658875	0.008974656508	0.009300812234
# 60	0.009579542766	0.008953002696	0.009411337640

When we run test set with the tuned RF the RMSE for RF is 0.1930190968 which is worse than the RMSE of RR and LR. In training sets the RMSE's for RF are so good. And they are better than RR and LR. As we remember we use 2013 as our test data. And in 2013 there is a sharp decrease in prices. RF could not catch possible outcomes in training data and then behave so badly for 2013. In Figure 11 we can see the RF results, as you can see the predicted outcomes are higher in all days than the actual data. The green line is the actual values the red line is the predicted results. And the blue line is the difference between predicted – actual.

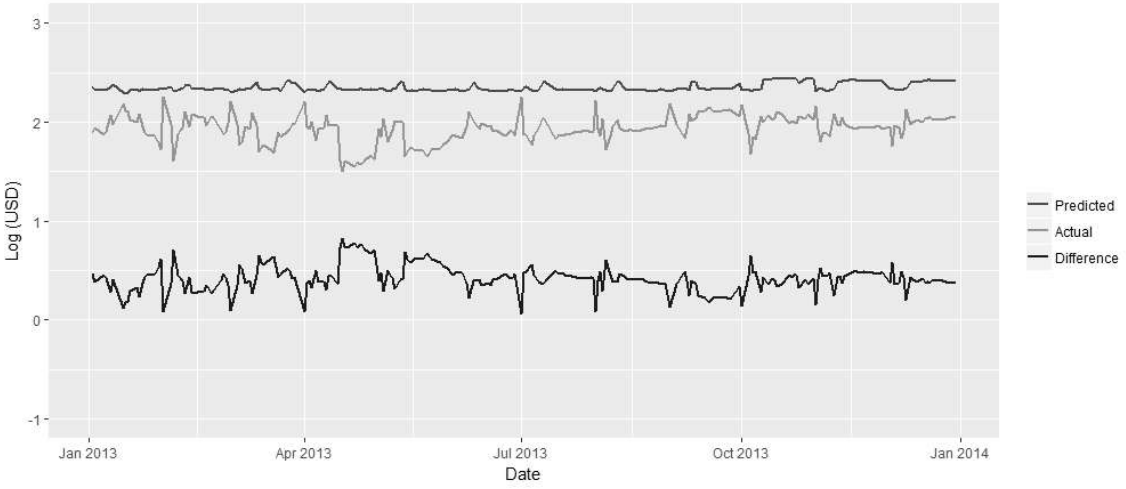

Figure 11: Predicted and Actual results for RF

At Figure 12 we can see the important features and their importance level for RF model.

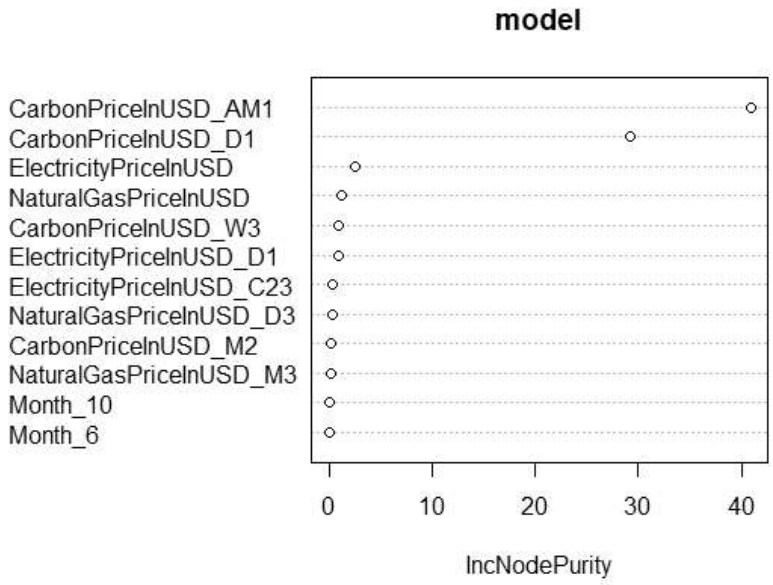

Figure 12: RF feature importance

4.1.5. Support Vector Machine (SVM) for First Scenario

In SVM, there are three parameters that affect model performance: cost, gamma and kernel type [2], [3]. For SVM parameter tuning we use "tune" [24] function. We use two types of kernels: Linear and Radial kernels. We firstly optimize linear SVM with 10-fold CV. We use cost values as c(0.001, 0.01, 0.1, 1,5). The optimal values for first linear SVM optimization are as in Table 17:

Table 17. Linear SVM grid search results

```
# Parameters:
#   SVM-Type:  eps-regression
# SVM-Kernel:  linear
# cost:  0.1
# gamma:  0.08333333333
# epsilon:  0.1
#
#
# Number of Support Vectors:  359
```

Then we use another 10-fold CV to find a better cost parameter. We change cost values that are close to previous optimal cost value. In our second case we took cost as c(0.05,0.1,0.2,0.3,0.5) which are close to 0.1. But here again our best model took the same result as in our first try.

The MSE for SVM is: 0.01670586365. In Figure 13, we can see the SVM results. The green line is the actual values the red line is the predicted results. And the blue line is the difference between predicted – actual.

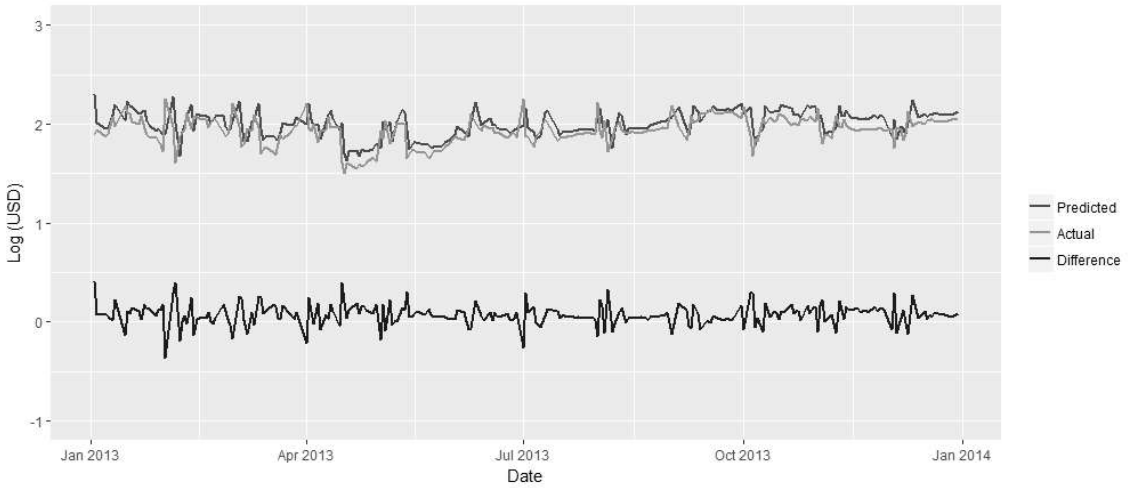

Figure 13: Predicted and Actual results for Linear SVM

Then we use another SVM with a Radial kernel. We use same methods for CV. But in that case the new RMSE is 0.6648747 which is worse of all. The optimized results for radial kernel are at Table 18.

Table 18. Radial SVM grid search results

```
# Parameters:
#   SVM-Type:  eps-regression
# SVM-Kernel:  radial
# cost:  2
# gamma:  0.5
# epsilon:  0.1
#
#
# Number of Support Vectors:  400
```

4.1.6. Artificial Neural Network (ANN) for First Scenario

In ANN, there are three layers. The input layers, hidden layers and the output layer [7]. Input layers are the n attributes used for model. Hidden layers and number of nodes in each layer is the critical part of ANN.

For ANN we do not use 10-fold CV, because CV for ANN is an expensive calculation and our resources are not enough for ANN CV. Then we use hidden layers as (8,6,4) as in the case of [5], in his experiment also used a hidden layer with layers (8,6,4) nodes respectively. We use “R” neural network package. That package uses scaled data. We use min max scaler before creating a model. We scale data with respect to train data set and use these scales for test data also.

After computing the predicted results, RMSE for ANN is 0.2561219732. In Figure 14 we can see the RF results, as you can see the predicted outcomes are higher in all days than the actual data. The green line is the actual values the red line is the predicted results. And the blue line is the difference between predicted – actual.

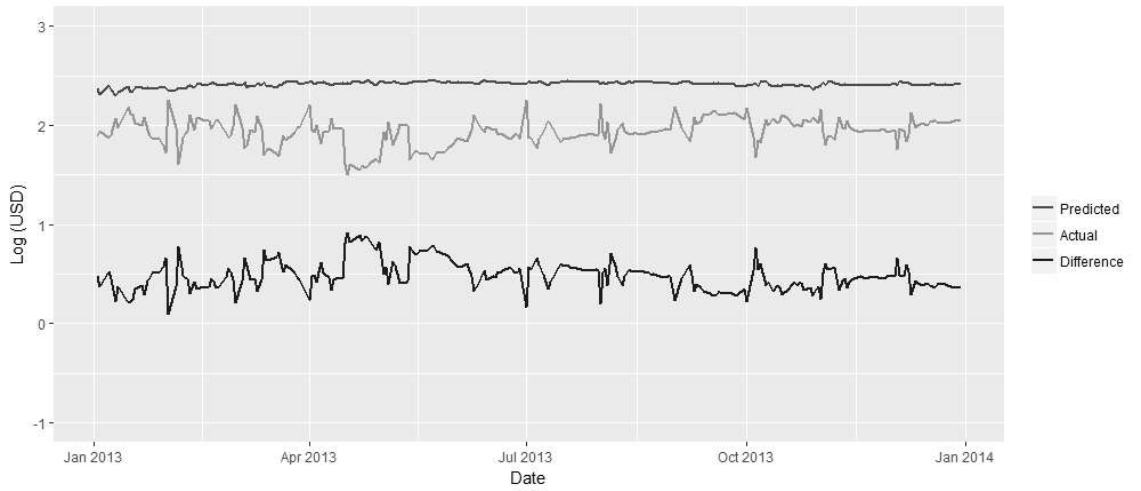

Figure 14: Predicted and Actual results for Linear ANN

In Figure 15, we show the ANN structure. The first 12 nodes are the input layers. The second 8 nodes are the first hidden layer, 6 nodes is the second hidden layer and 4 nodes are third hidden layer. The final unique node is the outcome node of the ANN.

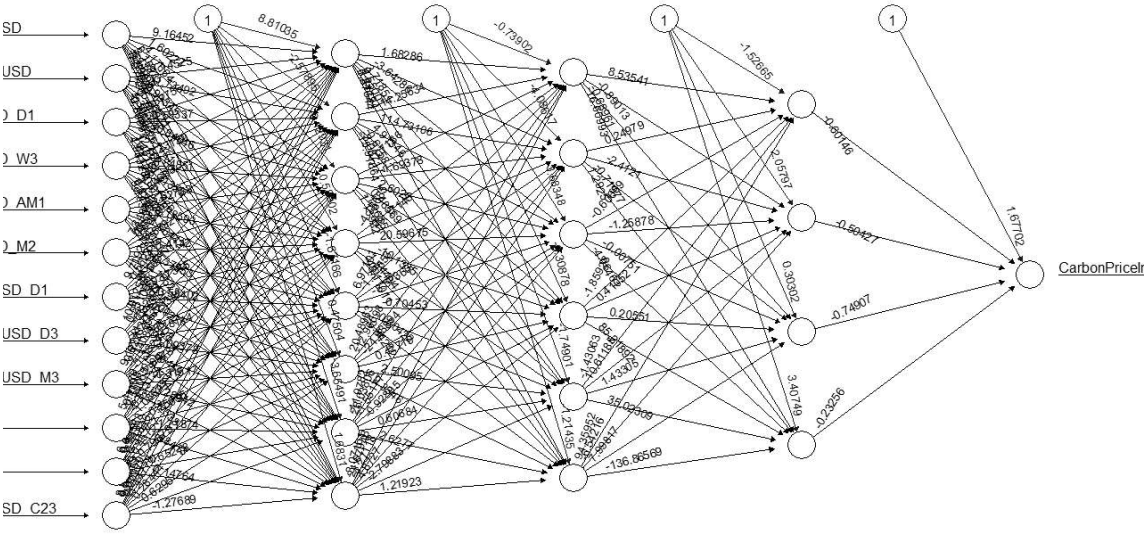

Figure 15: Layers of ANN

4.2. Second Scenario (Training Set: 2010-2014, Test Set: 2015)

For second scenario we do the same methods as in first scenario but in this case we use Year in (2010, 2011, 2012, 2013, 2014) for training set and 2015 for test set. In this scenario we expand our training set which will encounter more cases. So we expect some learning algorithms will give better results.

4.2.1. Feature Selection for Second Scenario

The attributes in Table 19 are eliminated in regression test. Those predictors are same as at Table 10, which are eliminated in scenario 1.

Table 19. Eliminated attributes because of correlation with target

```
#[1] "CarbonPriceInUSD_C12" "CarbonPriceInUSD_C23" "CoalPriceInUSD_C12"   "CoalPriceInUSD_C23"
```

Then we do again backward and forward selection for feature selection. The RMSE for forward selection is at Table 20. 20 predictors give the best RMSE. The minimum RMSE is 0.008386013302.

Table 20. RMSEs for forward selection and number of attributes

1	2	3	4	5	6	7	8	9	10	11
0.011571280	0.010102678	0.010078907	0.009472655	0.009123225	0.009079276	0.009109566	0.009118596	0.009044977	0.009069308	0.009055712
12	13	14	15	16	17	18	19	20	21	22
0.008994747	0.008896753	0.008810521	0.008688866	0.008709888	0.008620297	0.008513925	0.008506600	0.008386013	0.008410945	0.008427928
23	24	25	26	27	28	29	30	31	32	33
0.008423678	0.008445518	0.008458912	0.008455894	0.008508154	0.008492450	0.008500714	0.008517405	0.008524343	0.008542770	0.008510009
34	35	36	37	38	39	40	41	42	43	44
0.008493906	0.008485193	0.008464530	0.008471187	0.008475618	0.008536264	0.008553751	0.008530296	0.008528966	0.008542100	0.008572175
45	46	47	48	49	50	51	52	53	54	55
0.008557964	0.008555361	0.008563520	0.008539717	0.008529505	0.008523839	0.008539996	0.008554731	0.008579814	0.008606753	0.008638258
56	57	58	59	60	61	62	63	64	65	66
0.008655293	0.008654914	0.008680293	0.008672840	0.008689056	0.008682380	0.008684571	0.008685106	0.008686144	0.008693178	0.008700561
67	68	69	70	71	72	73	74	75	76	77
0.008696968	0.008685434	0.008697244	0.008686906	0.008699808	0.008679960	0.008678414	0.008668142	0.008673099	0.008706856	0.008707785
78	79	80	81	82	83	84	85	86	87	88
0.008705651	0.008722556	0.008721524	0.008718118	NA	NA	NA	NA	NA	NA	NA

From Figure 16, we see that when the number of attributes increases initially RMSE decreases sharply and it reaches the best value and then again increases.

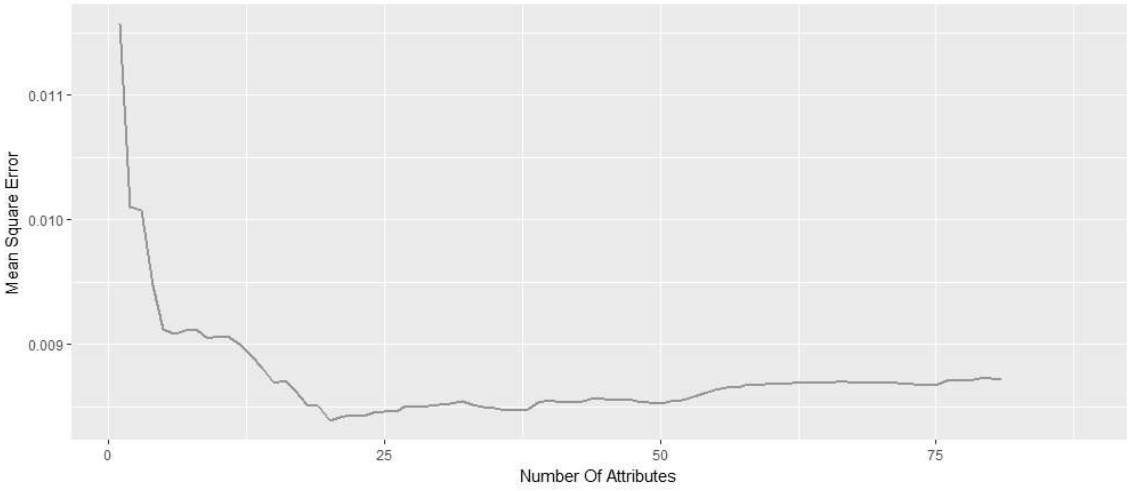

Figure 16: RMSE for forward selection and number of attributes

In Table 21, we list the best predictive useful features (20 features) for forward selection.

Table 21. Predictive features for forward selection

[1]	"CoalPriceInUSD"	"ElectricityPriceInUSD"	"NaturalGasPriceInUSD"	"CarbonPriceInUSD_D1"
[5]	"CarbonPriceInUSD_D3"	"CarbonPriceInUSD_W2"	"CarbonPriceInUSD_W3"	"CarbonPriceInUSD_AM1"
[9]	"CarbonPriceInUSD_M2"	"ElectricityPriceInUSD_D1"	"ElectricityPriceInUSD_W2"	"ElectricityPriceInUSD_W3"
[13]	"ElectricityPriceInUSD_AM1"	"ElectricityPriceInUSD_AM3"	"NaturalGasPriceInUSD_D1"	"NaturalGasPriceInUSD_D3"
[17]	"NaturalGasPriceInUSD_M3"	"Month_2"	"Month_6"	"Weekday_Mon"

Then we do backward feature section. The RMSE for backward selection is at table 22. 70 predictors give the best RMSE. The minimum RMSE is with 0.008673742579.

Table 22. RMSEs for backward selection and number of attributes

1	2	3	4	5	6	7	8	9	10	11
0.011571280	0.010797685	0.010623347	0.010396863	0.010319061	0.010105661	0.009932919	0.009688167	0.009773065	0.009500893	0.009401005
12	13	14	15	16	17	18	19	20	21	22
0.009366652	0.009332019	0.009344719	0.009288503	0.009307025	0.009318238	0.009278057	0.009233947	0.009235529	0.009189438	0.009215614
23	24	25	26	27	28	29	30	31	32	33
0.009217884	0.009209768	0.009146905	0.009099858	0.009082996	0.009071746	0.009062986	0.009034737	0.009036447	0.009055635	0.009051223
34	35	36	37	38	39	40	41	42	43	44
0.009036225	0.009028727	0.008963883	0.008970708	0.008944704	0.008976520	0.008980053	0.008942172	0.008846284	0.008860584	0.008898257
45	46	47	48	49	50	51	52	53	54	55
0.008885489	0.008926415	0.009014284	0.008838960	0.008870886	0.008866418	0.008892404	0.008833987	0.008836343	0.008829731	0.008807402
56	57	58	59	60	61	62	63	64	65	66
0.008717482	0.008747778	0.008753716	0.008767748	0.008730750	0.008725513	0.008732143	0.008727982	0.008716319	0.008736922	0.008756972
67	68	69	70	71	72	73	74	75	76	77
0.008688537	0.008690084	0.008682065	0.008673743	0.008676702	0.008683171	0.008685878	0.008699490	0.008692914	0.008696656	0.008701357
78	79	80	81	82	83	84	85	86	87	88
0.008716568	0.008715182	0.008724000	0.008724230	NA	NA	NA	NA	NA	NA	NA

In Figure 17, we see the backward selection results.

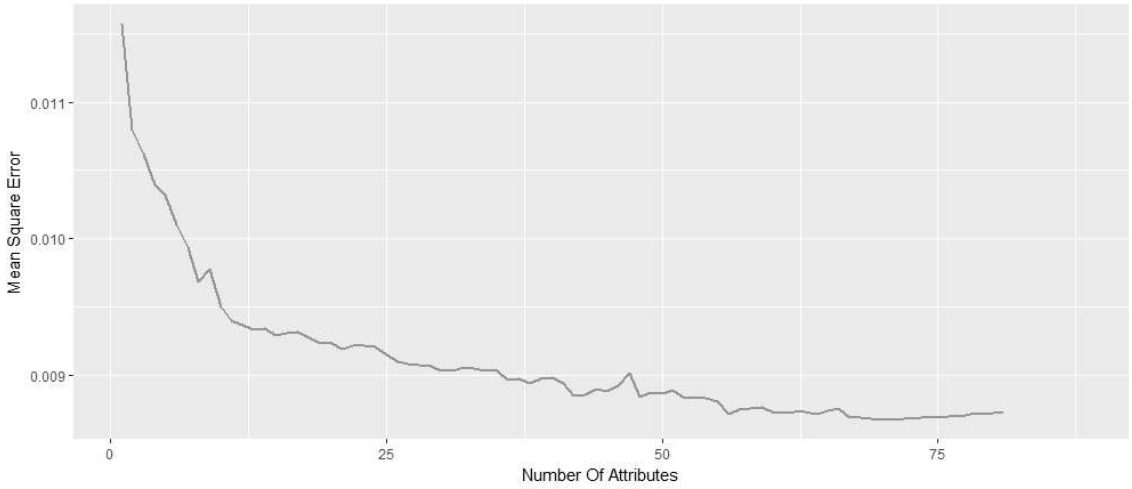

Figure 17: RMSE for backward selection and number of attributes

The most predictive features (70 features) for backward selection are at table 23.

Table 23. Predictive features for backward selection

[1] "Date"	"CoalPriceInUSD"	"ElectricityPriceInUSD"	"NaturalGasPriceInUSD"
[5] "CarbonPriceInUSD_D1"	"CarbonPriceInUSD_D2"	"CarbonPriceInUSD_D3"	"CarbonPriceInUSD_CR12"
[9] "CarbonPriceInUSD_W1"	"CarbonPriceInUSD_W2"	"CarbonPriceInUSD_W3"	"CarbonPriceInUSD_AM1"
[13] "CarbonPriceInUSD_M2"	"CarbonPriceInUSD_M3"	"CoalPriceInUSD_D1"	"CoalPriceInUSD_D2"
[17] "CoalPriceInUSD_D3"	"CoalPriceInUSD_CR12"	"CoalPriceInUSD_CR23"	"CoalPriceInUSD_W1"
[21] "CoalPriceInUSD_AW1"	"CoalPriceInUSD_W2"	"CoalPriceInUSD_M1"	"CoalPriceInUSD_AM1"
[25] "CoalPriceInUSD_M2"	"CoalPriceInUSD_M3"	"CoalPriceInUSD_AM3"	"ElectricityPriceInUSD_D1"
[29] "ElectricityPriceInUSD_D2"	"ElectricityPriceInUSD_D3"	"ElectricityPriceInUSD_CR23"	"ElectricityPriceInUSD_AW1"
[33] "ElectricityPriceInUSD_W2"	"ElectricityPriceInUSD_W3"	"ElectricityPriceInUSD_AM1"	"ElectricityPriceInUSD_M2"
[37] "ElectricityPriceInUSD_M3"	"ElectricityPriceInUSD_AM3"	"NaturalGasPriceInUSD_D1"	"NaturalGasPriceInUSD_D2"
[41] "NaturalGasPriceInUSD_CR23"	"NaturalGasPriceInUSD_AW1"	"NaturalGasPriceInUSD_W3"	"NaturalGasPriceInUSD_M1"
[45] "NaturalGasPriceInUSD_AM1"	"NaturalGasPriceInUSD_M2"	"NaturalGasPriceInUSD_M3"	"NaturalGasPriceInUSD_AM3"
[49] "Year"	"Month_11"	"Month_12"	"Month_2"
[53] "Month_3"	"Month_4"	"Month_5"	"Month_6"
[57] "Month_7"	"Month_8"	"Month_9"	"Weekday_Mon"
[61] "Weekday_Sat"	"Weekday_Sun"	"Weekday_Thu"	"Weekday_Wed"
[65] "Season_Spring"	"Season_Summer"	"Season_Winter"	"ElectricityPriceInUSD_C12"
[69] "Quarter_Fourth"	"Quarter_Third"		

RMSE for forward selection 0.008386013302 is less than RMSE for backward selection 0.008673742579. So we choose forward selection with 20 predictors for our models. The used predictors for models are in Table 21.

4.2.2. Ridge Regression (RR) for Second Scenario

The best lambda value for RR is 0.04681332, we use that value in prediction part. For model creation part, we use lambda as a sequence from 10^{10} to 10^{-2}.

When we run test set with the tuned RR the RMSE for RR is 0.002924469. In Figure 18, the green line is the actual values the red line is the predicted results. And the blue line is the difference between predicted – actual.

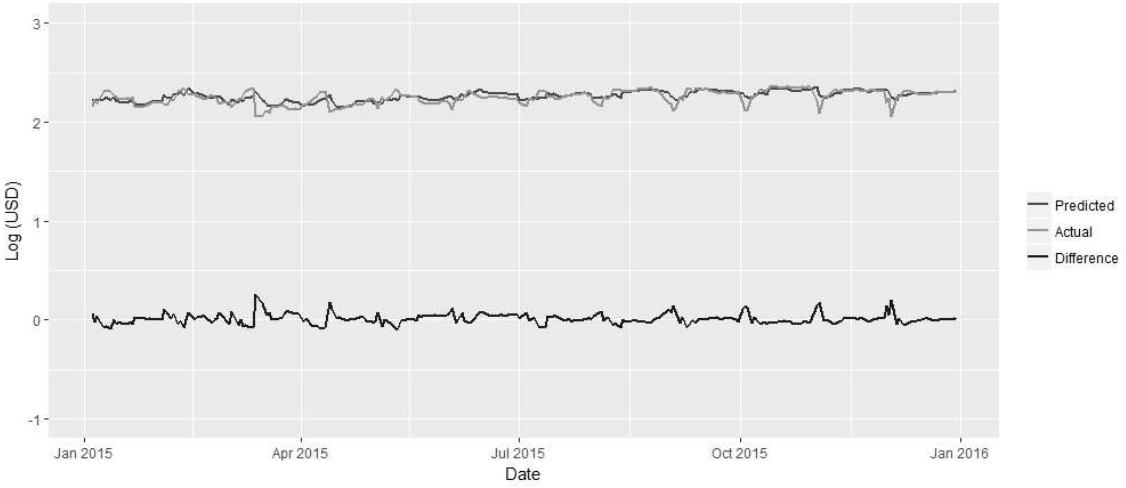

Figure 18: Predicted and Actual results for RR

4.2.3. Lasso Regression (LR) for Second Scenario

The best lambda value for LR is 0.0001889862, we use that value in prediction part. For model creation part, we used lambda as a sequence from 10^{10} to 10^{-2}.

When we run test set with the tuned LR the RMSE for LR is 0.00226574 which is better than the RMSE for RR 0.002924469. In Figure 19, we can see the LR results. The green line is the actual values the red line is the predicted results. And the blue line is the difference between predicted – actual.

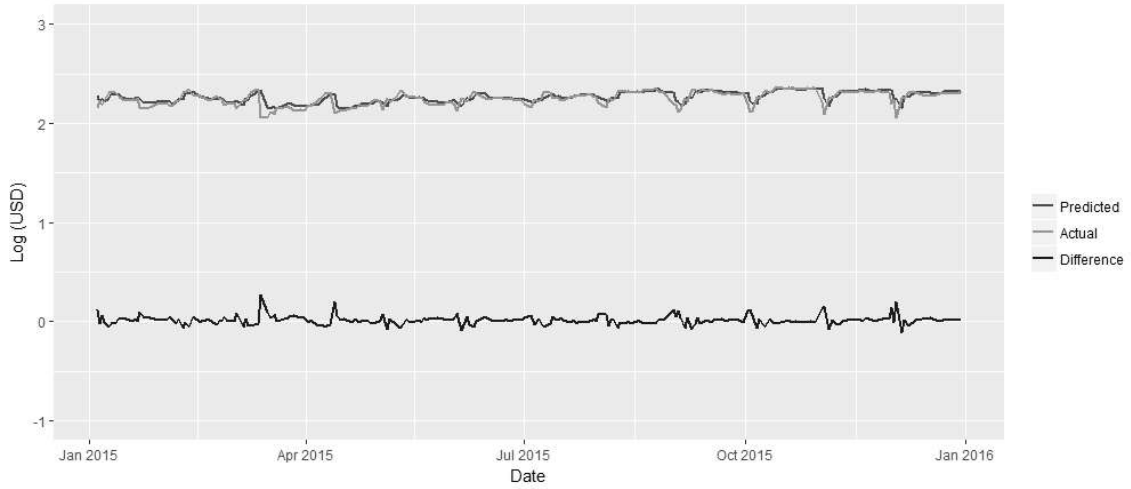

Figure 19: Predicted and Actual results for LR

4.2.4. Random Forest (RF) for Second Scenario

We use again CV to find optimal values for RF. We take mtree (number of tree) as c(50,100,500) and mtry (number of candidate predictors) as c(3,5,10). In Table 24, we can see the overall RMSEs for RF. The minimum RMSE for training set is 0.01103050. That means mtree=500 and mtry=10 gives the best result.

Table 24. RF RMSE results for grid search

mtry	3	5	10
mtree			
# 50	0.01489912	0.01185736	0.01189444
# 100	0.01424577	0.01192615	0.01151370
# 500	0.01374540	0.01185453	0.01103050

When we run test set with the tuned RF the RMSE for RF is 0.003152316 and has very good result then RMSE of our first scenario 0.2048478493. In Figure 20; we can see the

RF results, as you can see the predicted outcomes now fit better. The green line is the actual values the red line is the predicted results. And the blue line is the difference between predicted – actual.

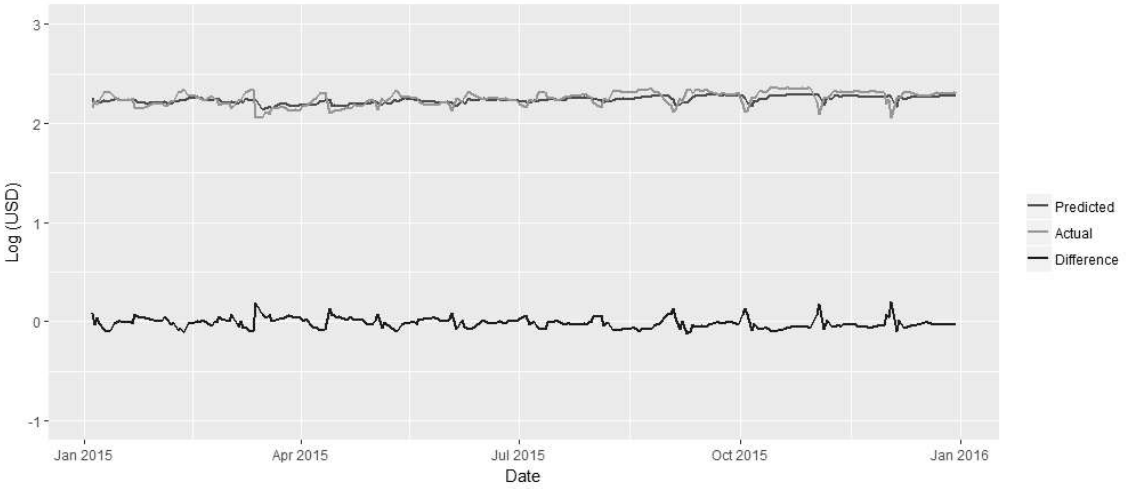

Figure 20: Predicted and Actual results for RF

The important parameters for RF are in Figure 21.

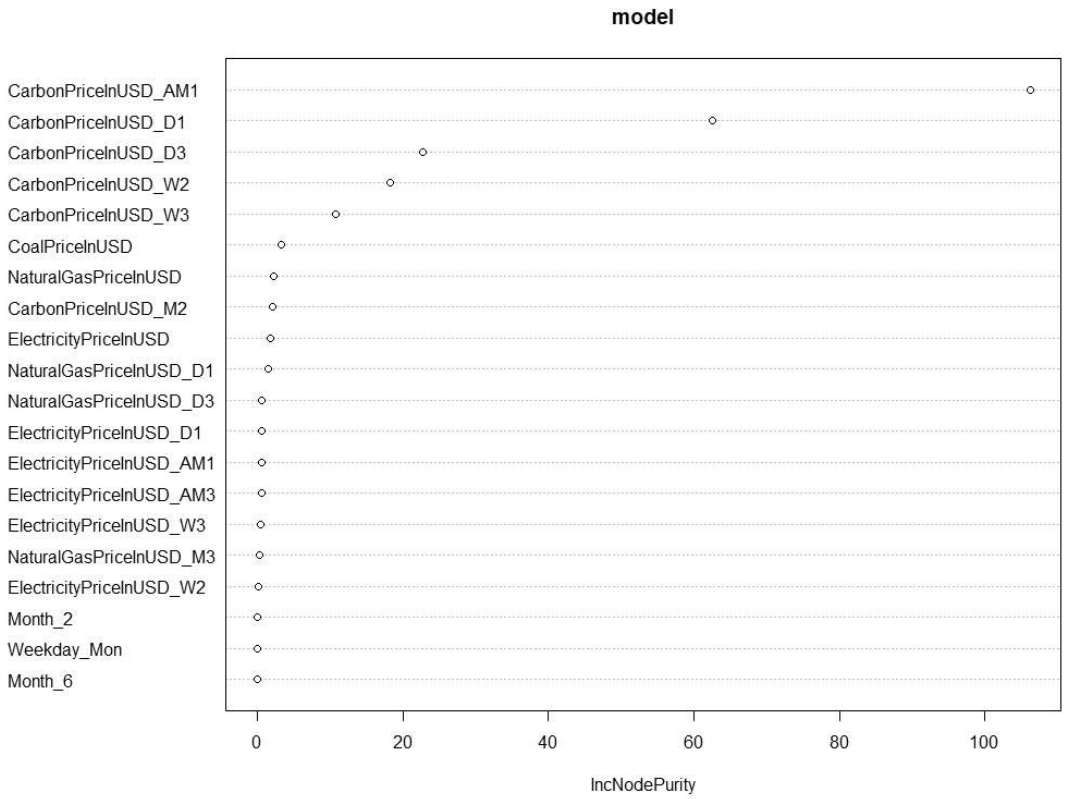

Figure 21: RF feature importance

4.2.5. Support Vector Machine (SVM) for Second Scenario

For SVM, we here again use two types of kernel: linear and radial kernel. The optimal values for linear SVM optimization are as in Table 25:

Table 25. Linear SVM grid search results

```
# Parameters:
#   SVM-Type:  eps-regression
# SVM-Kernel:  linear
# cost:  5
# gamma:  0.05
# epsilon:  0.1
#
#
# Number of Support Vectors:  524
```

By using optimal SVM with linear kernel we get MSE 0.00259102 for test data. In Figure 22, we can see the SVM results. The green line is the actual values the red line is the predicted results. And the blue line is the difference between predicted – actual.

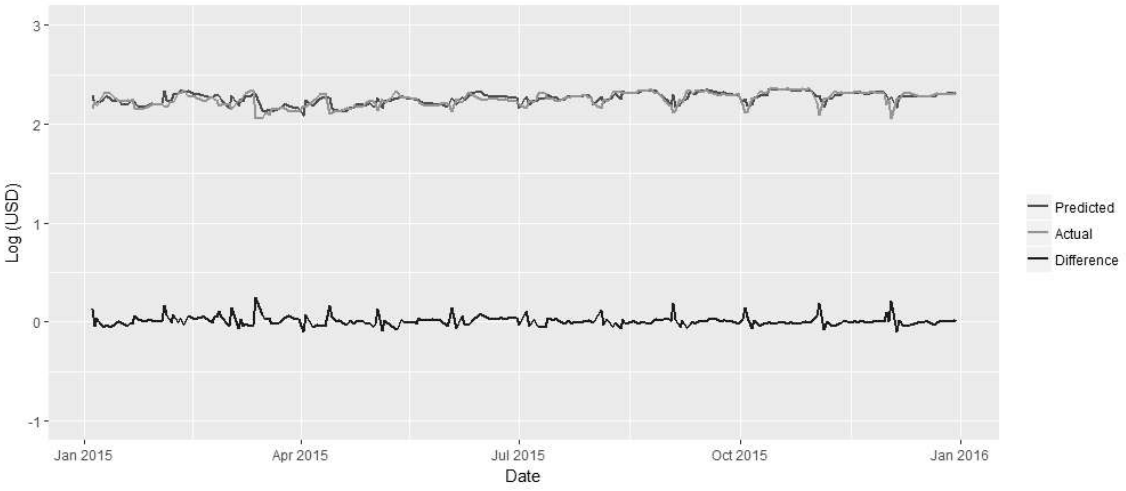

Figure 22: Predicted and Actual results for SVM

For radial kernel, we GET the results in table 26. RMSE for radial kernel is: 0.04221082312

Table 26. Radial SVM grid search results

```
# Parameters:
#   SVM-Type:  eps-regression
# SVM-Kernel:  radial
# cost:  4
# gamma:  0.5
# epsilon:  0.1
#
#
# Number of Support Vectors:  805
```

4.2.6. Artificial Neural Network (ANN) for Second Scenario

We use again hidden layers as (8,6,4) as in the case of [16]. For ANN, we firstly scale data with respect to train data set and use these scale ratios for test data. RMSE for ANN is 0.007977745443. In figure 23, we can see the ANN results The green line is the actual values

the red line is the predicted results. And the blue line is the difference between predicted – actual.

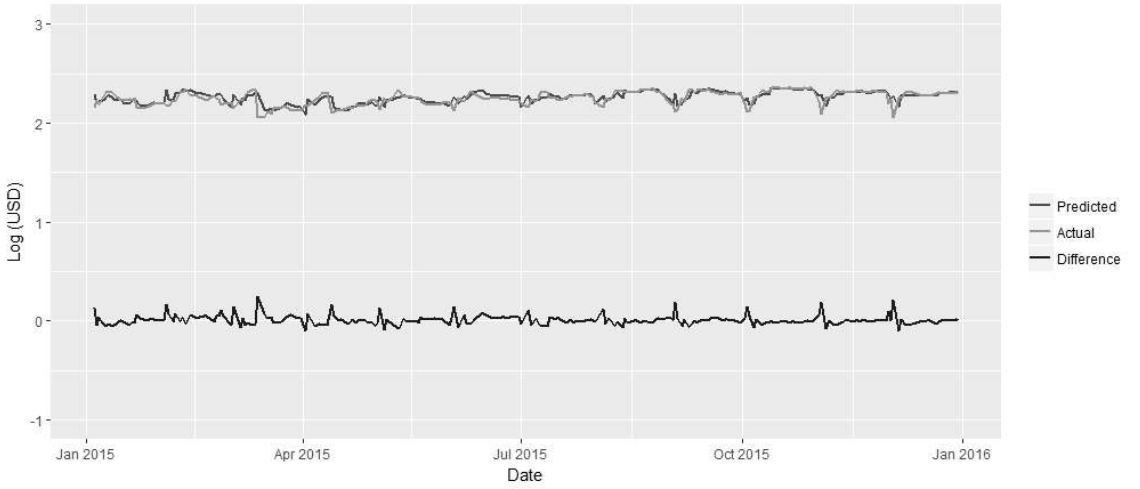

Figure 23: Predicted and Actual results for ANN

In Figure 24, we show the ANN structure. The first 20 nodes are the input layers. The second 8 nodes are the first hidden layer, 6 nodes is the second hidden layer and 4 nodes are third hidden layer. The final unique node is the outcome node of the ANN.

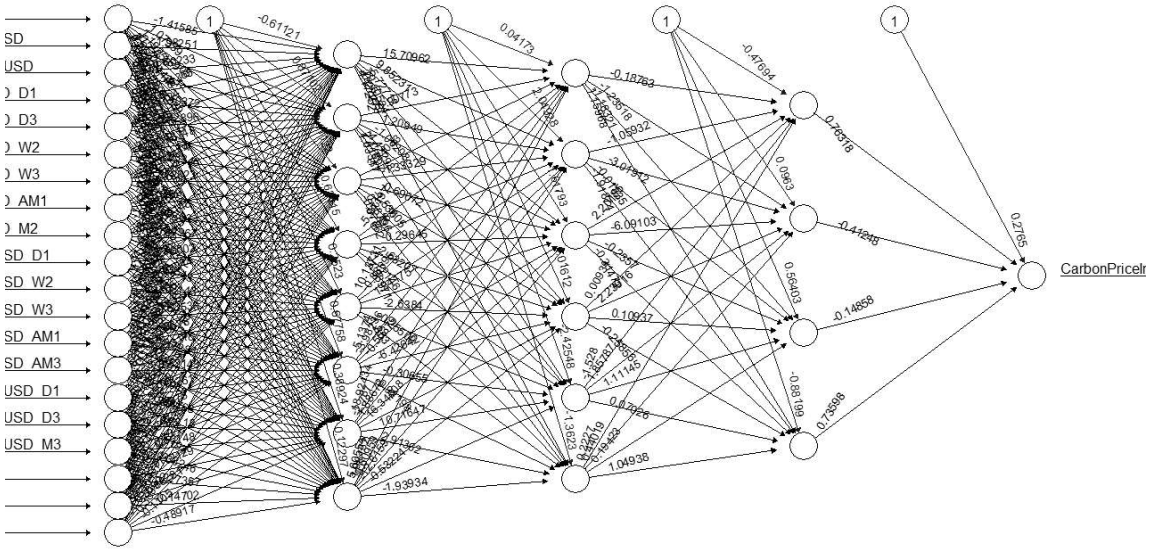

Figure 24: ANN layers

5. RESULTS

In this project, we use Carbon prices that are traded under European Union's Emissions Trading System (EU ETS). We forecast carbon prices by using carbon price data and some other sources. In the project, we have four different price sources which are: carbon, coal, natural gas and electricity. By using these kinds of sources and Date feature at section 3, we create new attributes. Then we take logarithm for numeric features. This project is a regression project. For all of categorical variables, at section 3 we create dummy variables. After creating log transformed numeric variables and dummy variables, we try to forecast carbon prices by using different machine learning methods. In our project, we use SVM, RF, LR, RR and ANN. At section 4, we first of all use grid search algorithms to select most predictive features. In our project, we try to understand how forecasting is affected by the amount of data. So, we create two scenarios for forecasting. In scenario 1, we have three years for training and one year for testing. In scenario 2, we have five years for training and 1 year for testing. After that we run five machine learning methods for those two scenarios. In our project, we use RMSE for comparing model performance.

We start with RR for both scenerios. RR results for both scenarios are good. In figure 24, we see the RR rsults for scenario 1. From figure 24, we see that RR results are close to actual values.

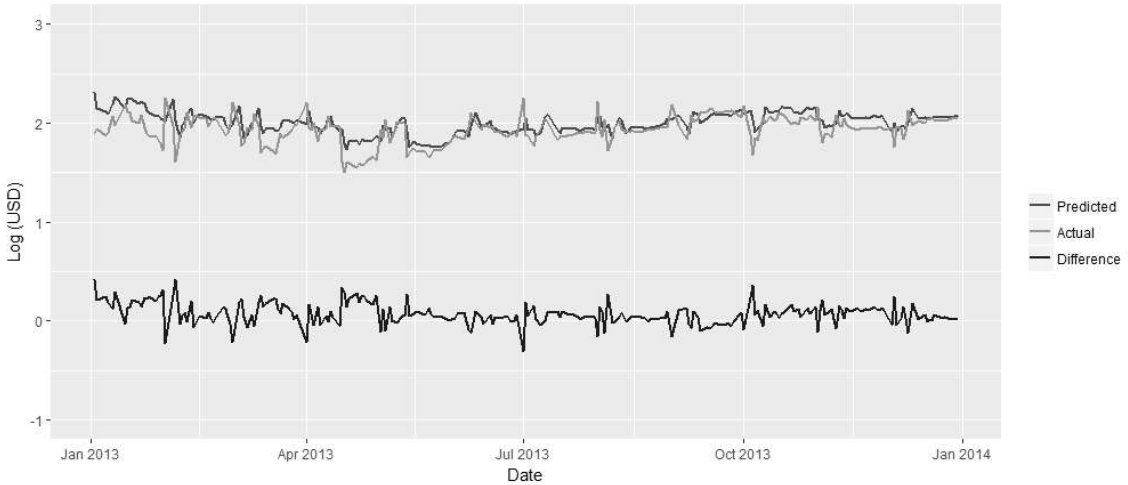

Figure 24. RR results for scenario 1

In figure 25, we see the RR results for scenario 2. We see form figure 25 that RR predicted values are more close to actual values for scenario 2. The difference are close to zero.

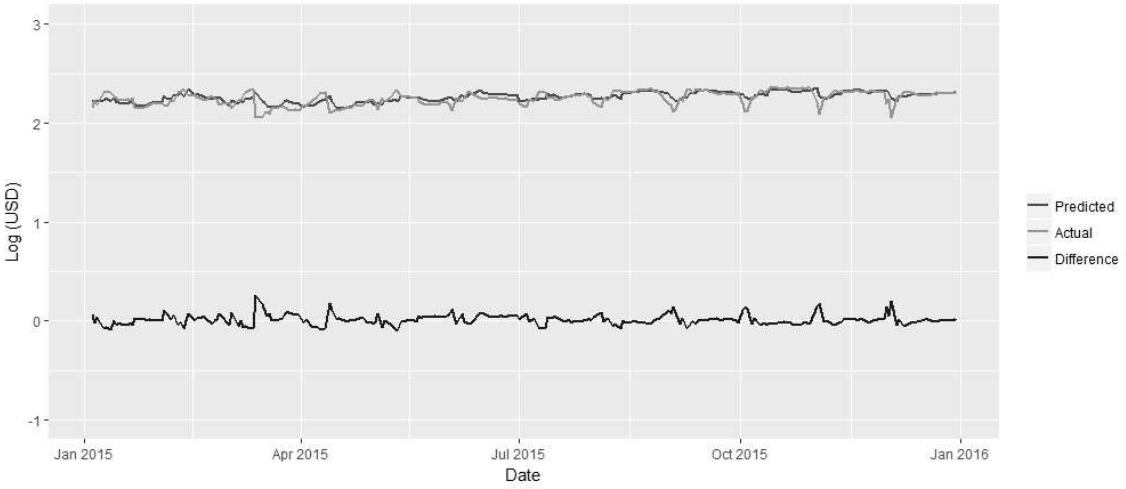

Figure 25. RR results for scenario 2

After RR we use LR for forecasting. In figure 26, we see the LR results for scenario 1. LR predicted values are close to actual values. LR gives better results for scenario 1 than RR.

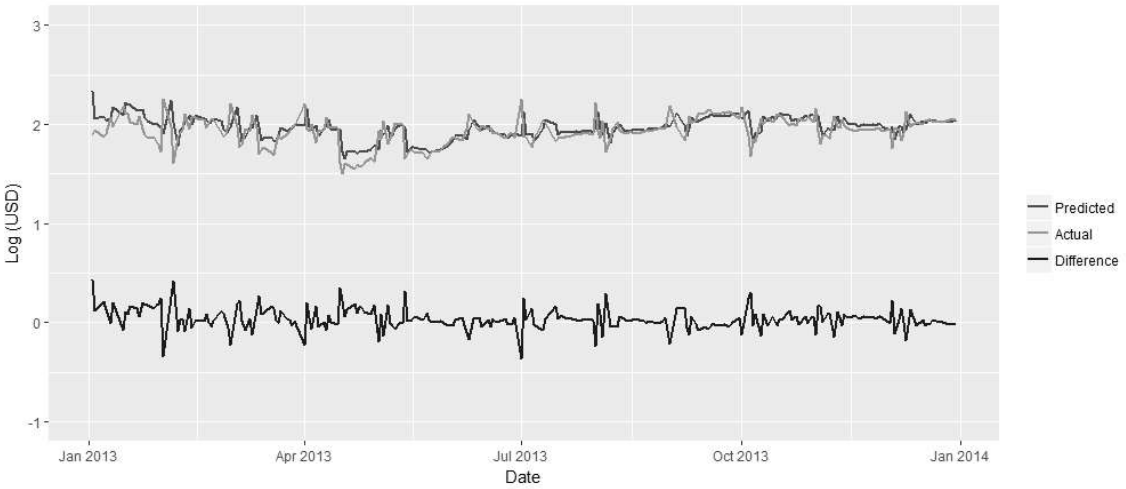

Figure 26. LR results for scenario 1

In figure 27, we see the LR results for scenari 2. LR results for scenario 2 gives the best results. The actual and predicted values are nearly same. From figure 27, we see that they are so close.

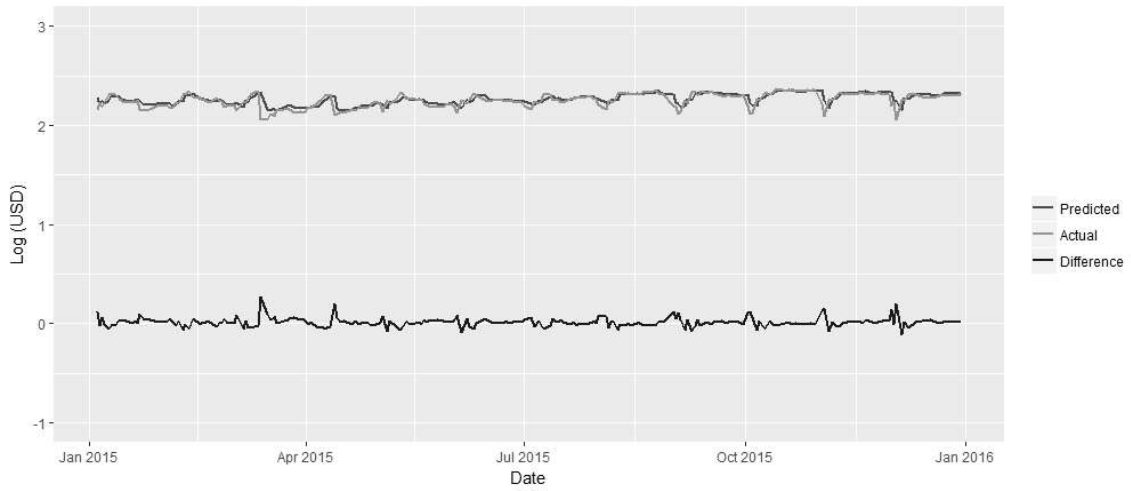

Figure 27. LR results for scenario 2

We use RF as our thid model for forecasting. In figure 28, we see the RF results for scenario 1. The predicted values for RF gives bigger results than the actual values for all cases in scenario 1.

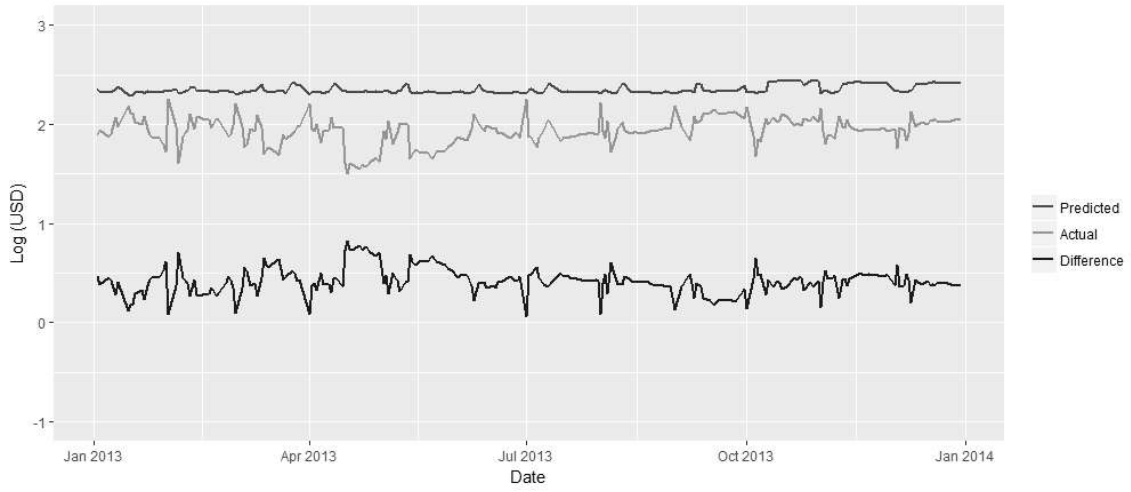

Figure 28. RF results for scenario 1

In figure 29, we show the RF results for scenario 2. In scenario 2, RF gives better results. In that case, RF sometimes gives smaller results than actual values. The predicted values for RF in scenario 2 are closer than the predicted values in scenario 1.

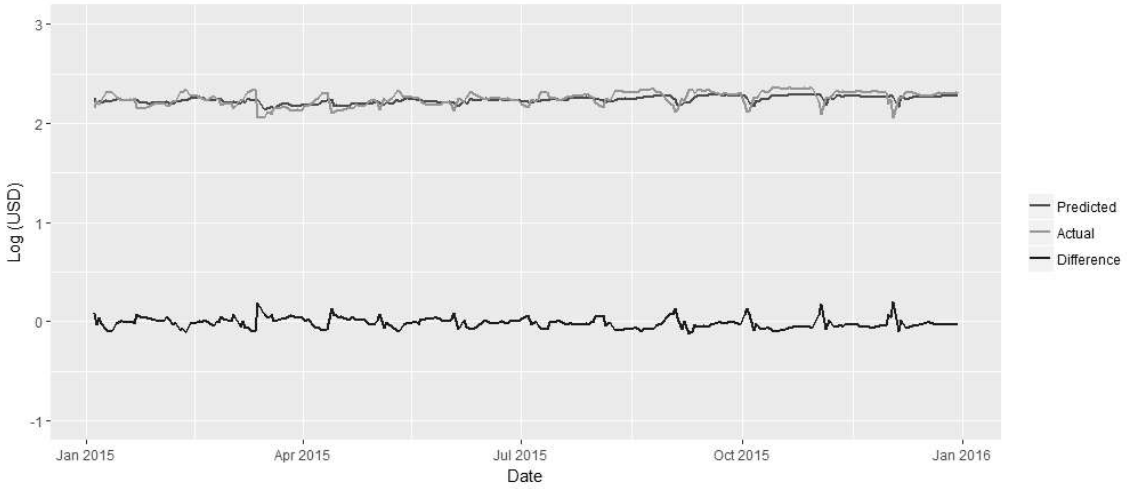

Figure 29. RF results for scenario 2

SVM is our fourth model for forecasting. In figure 30, we show the SVM results for scenario 1. The predicted values are close to actual values. SVM gives better results for scenario 1 than RF.

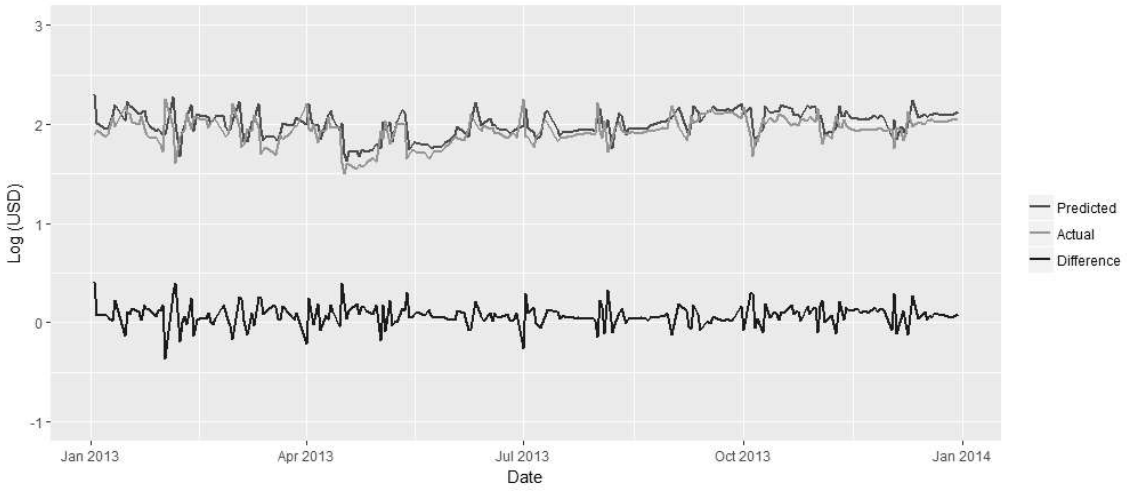

Figure 30. SVM results for scenario 1

In figure 31, we show the SVM results for scenario 2. In scenario 2, SVM gives better results. The predicted values for SVM in scenario 2 are closer than the predicted values in scenario 1. Here also, the actual and the predicted values are nearly same.

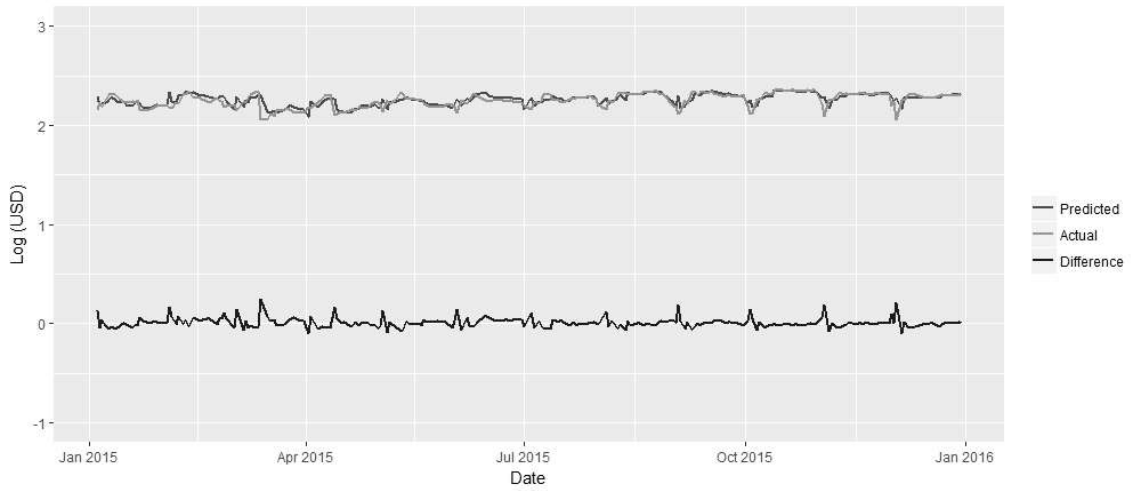

Figure 31. SVM results for scenario 2

We use ANN as our last model for forecasting. In figure 32, we see the ANN results for scenario 1. The predicted values for ANN gives bigger results than the actual values for all cases in scenario 1.

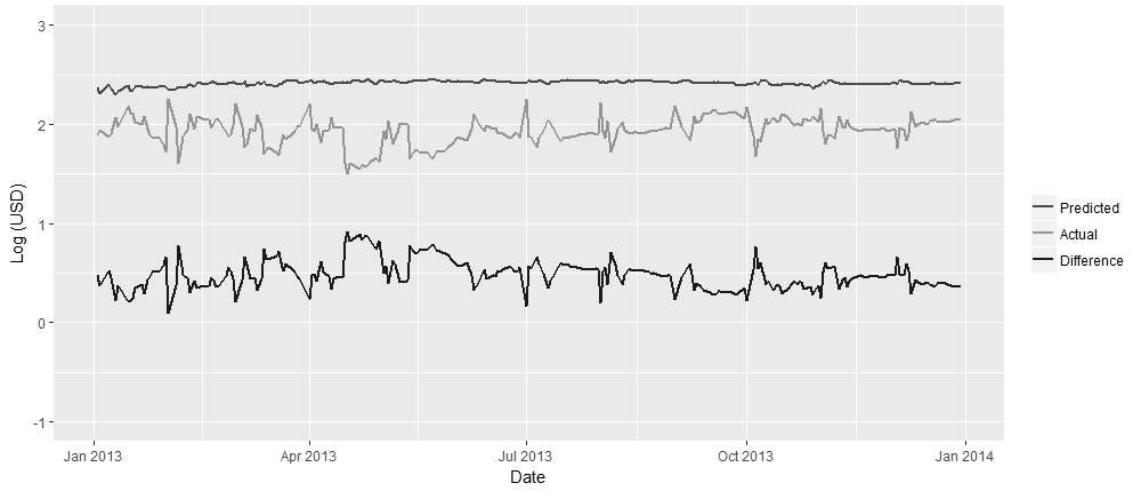

Figure 32. ANN results for scenario 1

In figure 33, we show the ANN results for scenario 2. In scenario 2, ANN gives better results. In that case, ANN sometimes gives smaller results than actual values. The predicted values for ANN in scenario 2 are closer than the predicted values in scenario 1.

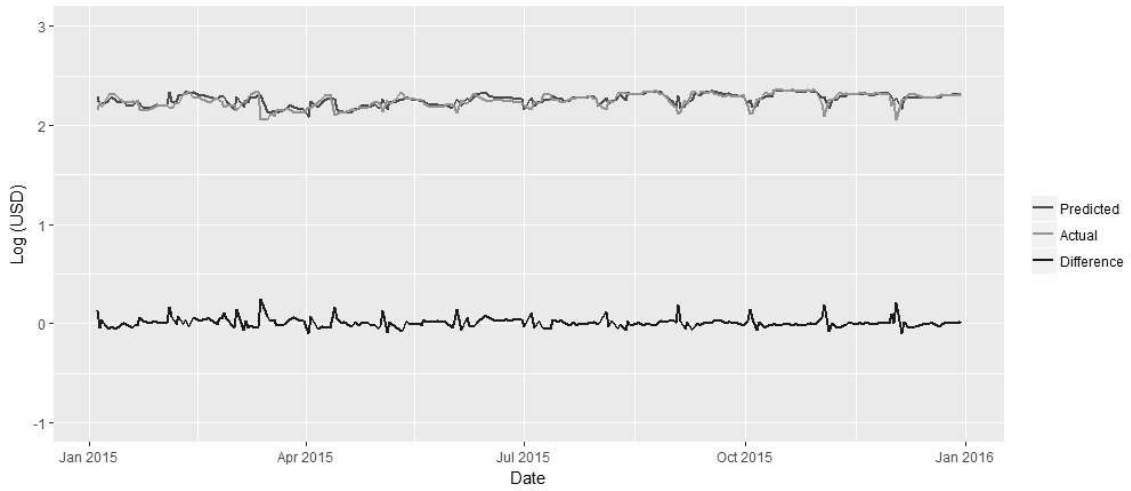

Figure 33. ANN results for scenario 2

When we combine all results and check their RMSE, we see that in all cases the second scenario has better RMSEs. It is because in second scenario the training set has more cases to learn. Most probably at first scenario the models were in biased situation. In second scenario the training data contained more cases and models learnt more. In both cases Lasso regression gives better results. But Ridge and SVM result are also close to the Lasso results. In Figure 34, we see the model performances. We see that RR, LL and SVM results are close to actual data. On the other hand, ANN and RF results have higher values for all dates.

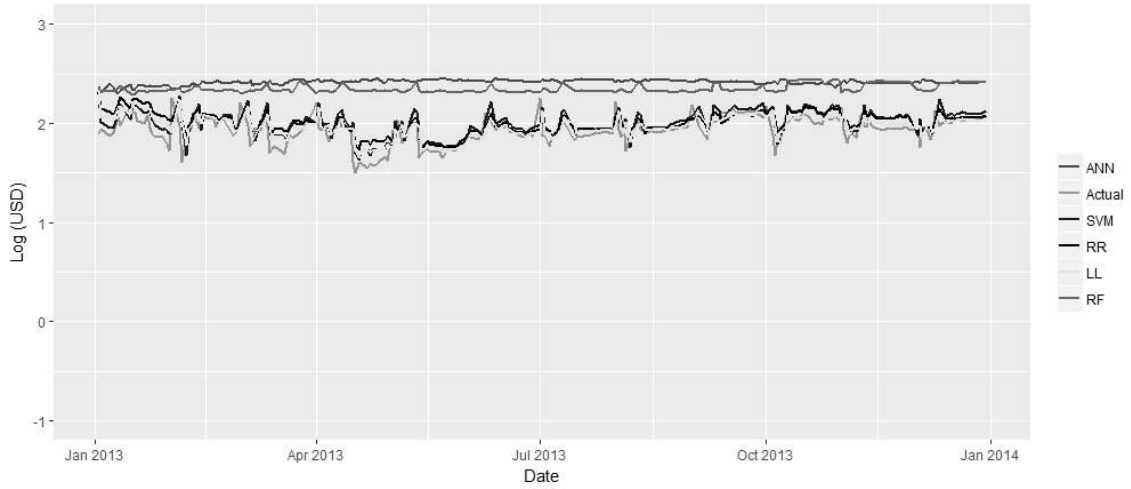

Figure 34: All model results for scenario 1

In Figure 35, we see the model results and actual data set. In scenario 2, all model give good results and they are close to actual data.

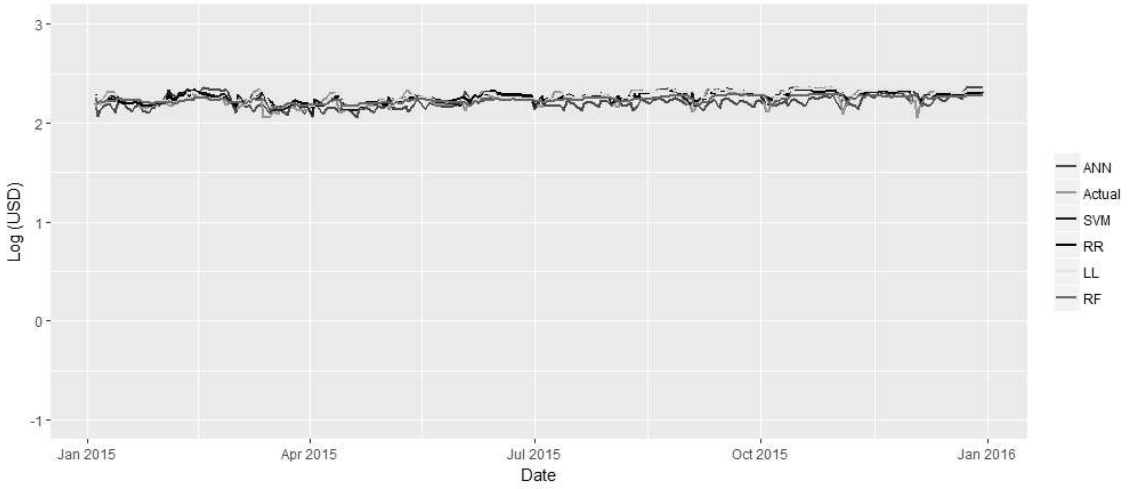

Figure 35: All model results for scenario 2

When we compare model performance with respect to RMSE, we see that for two scenario LS gives better results. We can see the all RMSE at Table 27.

Table 27: RMSE results for all models.

	First Scenario	Second Scenario
Ridge Regression	0.01698475748	0.002924469
Lasso Regression	0.01350746101	0.00226574
Random Forest	0.1930190968	0.003152316
Support Vector Mechine (Linear Kernel)	0.01670586365	0.00259102
Support Vector Mechine (Radial Kernel)	0.6648747	0.04221082312
Neural Network	0.2561219732	0.007977745443

6. FURTHER RESEARCH

In this project, we forecast carbon prices by using different machine learning methods. We use five different machine learning approaches for our project. In this project, we just use four prices: carbon price, natural gas price, electricity price and coal price. In section 3, we create new attributes by using those four main attributes and Date attribute. It is possible to create new attributes that are not in our list. For future experiments trend attributes in Table 28 can be used.

Table 28: Trend Attributes for Carbon Price.

Predictor Name	Description
CarbonPriceInUSD_NUM_INC_LW	Number of days in last week (last 7 days) that carbon price increases
CarbonPriceInUSD_NUM_INC_LM	Number of days in last month (last 30 days) that carbon price increases
CarbonPriceInUSD_NUM_DEC_LW	Number of days in last week (last 7 days) that carbon price decreases
CarbonPriceInUSD_NUM_DEC_LM	Number of days in last month (last 30 days) that carbon price decreases
CarbonPriceInUSD_MAX_CONS_INC_LW	Maximum number of consecutive days in last week (last 7 days) that carbon price increases
CarbonPriceInUSD_MAX_CONS_INC_LM	Maximum number of consecutive days in last month (last 30 days) that carbon price increases
CarbonPriceInUSD_MAX_CONS_DEC_LW	Maximum number of consecutive days in last week (last 7 days) that carbon price decreases
CarbonPriceInUSD_MAX_CONS_DEC_LM	Maximum number of consecutive days in last month (last 30 days) that carbon price decreases

We take prices as its current date we do not take into account USD value changes. For future researches, it can be better to take into account USD value changes. In that case, we will convert all prices with respect to end of 2017 USD exchange rates.

In our project, we do not take into account economical crises in our data sets. For future experiments, it can be better to include important event dates for forecasting. Econimical and envireemantal crises, political problems between countries can be a good predictive attribute for forecasting.

In our project, we use five different machine learning methods: LR, RR, RF, SVM and ANN. There are also other popular machine learning methods. It can be better to include new methods like Xgboost [26] for future experiments.

7. CONCLUSION

In this project, we use different machine learning methods to forecast carbon prices. In our data set we combine different sources to create a unique data set. Our data set includes carbon and other energy resources. For modeling, we create two types of data sets: one includes three years for training and one year for testing, and the other one includes five years for training and one year for testing.

We use five machine learning methods: LR, RR, RF, SVM and ANN. In our experiment, LR gives the best results for two scenarios. Moreover; LR, RR and SVM give good results for this project and their predictions are close to actual values. On the other hand; in scenario 1, ANN and RF give higher predictions than the actual values. For scenario 2, all of five models give better results and their predicted values are close to actual values.

Our experiments show that:

- If the amount of training data increases all of the five models give better results.
- LR gives better results for two scenarios
- The prediction of LR, RR and SVM are close to actual values for both scenarios
- ANN and RF for our project are extremely affected by amount of training data

8. APPENDIX A: PYTHON DATA ANALYSIS CODE

```
#Nurhak Karakaya
#First part of Capstone project
#Data cleaning and preparation part

import os
import warnings
from time import time, sleep  # to keep track of the processing time
import random
import pandas as pd
import numpy as np
from datetime import datetime, timedelta

withoutCurrency = True

warnings.filterwarnings("ignore")

os.chdir('C:/dersler/capstone')
#os.listdir()

arr_txt = [x for x in os.listdir() if x.endswith(".csv")]
print(arr_txt)
#['carbon2013.csv', 'carbon2017.csv', 'coal.csv', 'elec.csv',
'eurousd.csv', 'gbpeuro.csv', 'gbpusd.csv', 'ngas.csv', 'usdeuro.csv']


carbon2013File = "carbon2013.csv"
carbon2017File = "carbon2017.csv"
coalFile = "coal.csv"
elecFile = "elec.csv"
eurousdFile = "eurousd.csv"
gbpeuroFile = "gbpeuro.csv"
gbpusdFile = "gbpusd.csv"
ngasFile = "ngas.csv"
usdeuroFile = "usdeuro.csv"

def readFileAndReturnDataFrame(fileName,name):
    dataSet = pd.read_csv(fileName,sep=';',encoding="ISO-8859-
1",dtype={'Date': str,'PX_LAST': float})
    dataSet.name = name
    print(dataSet.name)
    print(dataSet["Date"].describe())

    dates = dataSet["Date"]
    dubs = dataSet[dates.isin(dates[dates.duplicated(keep=False)])]["Date"]
    print(dubs)

    dataSet['Date'] = pd.to_datetime(dataSet['Date'])
    return dataSet

#['carbon2013.csv', 'carbon2017.csv', 'coal.csv', 'elec.csv',
'eurousd.csv', 'gbpeuro.csv', 'gbpusd.csv', 'ngas.csv', 'usdeuro.csv']

carbon2013 = readFileAndReturnDataFrame(carbon2013File,'carbon2013')
carbon2017 = readFileAndReturnDataFrame(carbon2017File,'carbon2017')
coal = readFileAndReturnDataFrame(coalFile,'coal')
elec = readFileAndReturnDataFrame(elecFile,'elec')
```

```
eurousd = readFileAndReturnDataFrame(eurousdFile,'eurousd')
gbpeuro = readFileAndReturnDataFrame(gbpeuroFile,'gbpeuro')
gbpusd = readFileAndReturnDataFrame(gbpusdFile,'gbpusd')
ngas = readFileAndReturnDataFrame(ngasFile,'ngas')
usdeuro = readFileAndReturnDataFrame(usdeuroFile,'usdeuro')


#combine two carbon dataframe
frames = [carbon2013,carbon2017]
carbon = pd.concat(frames)
carbon["Date"].describe()


temp = carbon.merge(eurousd,left_on="Date",right_on="Date",how="inner")
capstoneDataSet = temp.sort_values("Date")


#we merge carbon and erurousd
capstoneDataSet = capstoneDataSet.rename(columns={'PX_LAST_x':
'CarbonOriginalPrice', 'PX_LAST_y': 'EuroUSD'})
capstoneDataSet["CarbonPriceInUSD"] =
capstoneDataSet["CarbonOriginalPrice"]  * capstoneDataSet["EuroUSD"]
capstoneDataSet =
capstoneDataSet.merge(coal,left_on="Date",right_on="Date",how="inner")
capstoneDataSet = capstoneDataSet.rename(columns={'PX_LAST':
'CoalPriceInUSD'})
capstoneDataSet = capstoneDataSet.sort_values("Date")
#we cmbine electrycity
capstoneDataSet =
capstoneDataSet.merge(elec,left_on="Date",right_on="Date",how="inner")
capstoneDataSet = capstoneDataSet.rename(columns={'PX_LAST':
'ElectricityOriginalPrice'})
capstoneDataSet = capstoneDataSet.sort_values("Date")
#we combine gdpusd
capstoneDataSet =
capstoneDataSet.merge(gbpusd,left_on="Date",right_on="Date",how="inner")
capstoneDataSet = capstoneDataSet.rename(columns={'PX_LAST': 'GdpUSD'})
capstoneDataSet = capstoneDataSet.sort_values("Date")
capstoneDataSet["ElectricityPriceInUSD"] =
capstoneDataSet["ElectricityOriginalPrice"]  * capstoneDataSet["GdpUSD"]
#we add eurogdp
capstoneDataSet =
capstoneDataSet.merge(gbpeuro,left_on="Date",right_on="Date",how="inner")
capstoneDataSet = capstoneDataSet.rename(columns={'PX_LAST': 'GdpEuro'})
capstoneDataSet = capstoneDataSet.sort_values("Date")
#we add ngas
capstoneDataSet =
capstoneDataSet.merge(ngas,left_on="Date",right_on="Date",how="inner")
capstoneDataSet = capstoneDataSet.rename(columns={'PX_LAST':
'NaturalGasOriginalPrice'})
capstoneDataSet["NaturalGasPriceInUSD"] =
capstoneDataSet["NaturalGasOriginalPrice"]  * capstoneDataSet["GdpUSD"]
capstoneDataSet = capstoneDataSet.sort_values("Date")

del capstoneDataSet["NaturalGasOriginalPrice"]
del capstoneDataSet["ElectricityOriginalPrice"]
del capstoneDataSet["CarbonOriginalPrice"]


lst = [carbon2013, carbon2017, carbon, coal, elec,eurousd, gbpeuro,
gbpusd,ngas, usdeuro,temp]
```

```
del carbon2013, carbon2017, carbon, coal, elec,eurousd, gbpeuro,
gbpusd,ngas, usdeuro,temp # dfs still in list
del lst     # memory release now

capstoneDataSet = capstoneDataSet.sort_values("Date")

#now i will clean null values from my data set
table_stats=pd.DataFrame(capstoneDataSet.dtypes).T.rename(index={0:'column
type'})
table_stats=table_stats.append(pd.DataFrame(capstoneDataSet.isnull().sum())
.T.rename(index={0:'null values (nb)'}))
table_stats=table_stats.append(pd.DataFrame(capstoneDataSet.isnull().sum()/
capstoneDataSet.shape[0]*100).T.
                         rename(index={0:'null values (%)'}))
table_stats=table_stats.append(pd.DataFrame(capstoneDataSet.nunique()).T.re
name(index={0:'Unique Count'}))

print(table_stats)# i dont hane any null values.
print('Dublicate Count:',capstoneDataSet.duplicated().sum())
#Dublicate Count: 39

#we can remove still remainig dublicates.
capstoneDataSet.drop_duplicates(inplace = True)

#a = capstoneDataSet
#capstoneDataSet = a
#we now find duplucates and fix them
dates = capstoneDataSet["Date"]
dubs =
capstoneDataSet[dates.isin(dates[dates.duplicated(keep=False)])].index
dubs

capstoneDataSet.drop(capstoneDataSet.index[dubs],inplace=True)



#CarbonPriceInUSD                      float64
def updatePreviousDays(column_name,first,last):
    for i in range(first, last+1):
        if (i > 1):
            capstoneDataSet[column_name + "_D1"][i] =
capstoneDataSet[column_name][i-1]

        if (i >= 2):
            capstoneDataSet[column_name + "_D2"][i] =
capstoneDataSet[column_name][i-2]
            capstoneDataSet[column_name +"_C12"][i] =
capstoneDataSet[column_name][i-1] - \

capstoneDataSet[column_name][i - 2]
            capstoneDataSet[column_name +"_CR12"][i] =
capstoneDataSet[column_name][i-1] / \

capstoneDataSet[column_name][i - 2]

        if (i >= 3):
            capstoneDataSet[column_name + "_D3"][i] =
capstoneDataSet[column_name][i-3]
            capstoneDataSet[column_name +"_C23"][i] =
capstoneDataSet[column_name][i-2] - \
```

```
capstoneDataSet[column_name][i - 3]
            capstoneDataSet[column_name +"_CR23"][i] =
capstoneDataSet[column_name][i-2] / \

capstoneDataSet[column_name][i - 3]

        weekSum = 0
        if (i >= 7):
            capstoneDataSet[column_name + "_W1"][i] =
capstoneDataSet[column_name][i-7]
            div = 7
            for j in range(1,8):
                weekSum = weekSum + capstoneDataSet[column_name][i-j]

            capstoneDataSet[column_name + "_AW1"][i] = weekSum / div

        if (i >= 14):
            capstoneDataSet[column_name + "_W2"][i] =
capstoneDataSet[column_name][i-14]
        if (i >= 21):
            capstoneDataSet[column_name + "_W3"][i] =
capstoneDataSet[column_name][i-21]

        monthSum = weekSum
        if (i >= 30):
            capstoneDataSet[column_name + "_M1"][i] =
capstoneDataSet[column_name][i-30]
            sm = 0
            div = 30
            for j in range(9,31):
                monthSum = monthSum + capstoneDataSet[column_name][i-j]
            capstoneDataSet[column_name + "_AM1"][i] = monthSum / div
        if (i >= 60):
            capstoneDataSet[column_name + "_M2"][i] =
capstoneDataSet[column_name][i-60]

        threemonthSum = monthSum
        if (i >= 90):
            capstoneDataSet[column_name + "_M3"][i] =
capstoneDataSet[column_name][i-90]
            sm = 0
            div = 90
            for j in range(32,91):
                threemonthSum = threemonthSum +
capstoneDataSet[column_name][i-j]

            capstoneDataSet[column_name + "_AM3"][i] = threemonthSum / div

#CarbonPriceInUSD                       float64

def create_new_column(df,column_name):
    list = [
        column_name + "_D1",
        column_name + "_D2",
        column_name + "_D3",
        column_name + "_C12",
        column_name + "_CR12",
        column_name + "_C23",
        column_name + "_CR23",
        column_name + "_W1",
        column_name + "_AW1",
```

```
        column_name + "_W2",
        column_name + "_W3",
        column_name + "_M1",
        column_name + "_AM1",
        column_name + "_M2",
        column_name + "_M3",
        column_name + "_AM3"
    ]
    df = df.reindex(columns=[*df.columns.tolist(), *list], fill_value=0.0)

    return df


list = [
    "CarbonPriceInUSD",
    "CoalPriceInUSD",
    "ElectricityPriceInUSD",
    "NaturalGasPriceInUSD",
    "EuroUSD",
    "GdpUSD",
    "GdpEuro"
]
listWithoutCurrency  = [
    "CarbonPriceInUSD",
    "CoalPriceInUSD",
    "ElectricityPriceInUSD",
    "NaturalGasPriceInUSD"
]

length = len(capstoneDataSet)
if (withoutCurrency):
    for column_name in listWithoutCurrency:
        capstoneDataSet = create_new_column(capstoneDataSet, column_name)

        first = 1
        last = 50

        while first < length:
            updatePreviousDays(column_name, first, last)
            print("First:", first)

            # wait for one second
            sleep(random.choice([0.1, 0.2, 0.3, 0.4]))
            first = last + 1
            last = last + 50

            if (last > length):
                last = length - 1

    del capstoneDataSet["GdpUSD"]
    del capstoneDataSet["EuroUSD"]
    del capstoneDataSet["GdpEuro"]

else:
    for column_name in list:
        capstoneDataSet = create_new_column(capstoneDataSet, column_name)

        first = 1
        last = 50

        while first < length:
```

```
            updatePreviousDays(column_name, first, last)
            print("First:", first)

            # wait for one second
            sleep(random.choice([0.1, 0.2, 0.3, 0.4]))
            first = last + 1
            last = last + 50

            if (last > length):
                last = length - 1

#capstoneDataSet['YearMonth'] = capstoneDataSet['Date'].apply(lambda x:
(100*x.year) + x.month)
capstoneDataSet['Month'] = capstoneDataSet['Date'].apply(lambda x: x.month)
capstoneDataSet['Year'] = capstoneDataSet['Date'].apply(lambda x: (x.year)
)
capstoneDataSet['Weekday'] = capstoneDataSet['Date'].apply(lambda x:
x.strftime('%a'))

capstoneDataSet["Weekday"].head(10)
def getQuarter(row):
    if row['Month'] in (1,2,3):
        return 'First'
    elif row['Month'] in (4,5,6):
        return 'Second'
    elif row['Month'] in (7,8,9):
        return 'Third'
    else:
        return 'Fourth'

capstoneDataSet['Quarter'] = capstoneDataSet.apply(getQuarter, axis=1)

capstoneDataSet.head()


def get_season(row):
    if (
            row["Date"].month in (7,8)
            or
            (row["Date"].month == 6 and row["Date"].day >= 21)
            or
            (row["Date"].month == 9 and row["Date"].day <= 22)
        ):
        return "Summer"
    elif (
            row["Date"].month in (10,11)
            or
            (row["Date"].month == 9 and row["Date"].day >= 23)
            or
            (row["Date"].month == 12 and row["Date"].day <= 20)
        ):
        return "Autumn"
    elif (
            row["Date"].month in (4,5)
            or
            (row["Date"].month == 3 and row["Date"].day >= 21)
            or
            (row["Date"].month == 6 and row["Date"].day <= 20)
        ):
        return "Spring"
    else:
```

```
        return "Winter"

capstoneDataSet['Season'] = capstoneDataSet.apply(get_season, axis=1)

#capstoneDataSet = capstoneDataSet.drop(['YearMonth'], inplace=True,
axis=1)

#save to file for later use
capstoneDataSet.columns
capstoneDataSet.shape
capstoneDataSet.dtypes
file_name = "capstone.csv"
if (withoutCurrency):
    file_name = "capstone_without_cur.csv"

capstoneDataSet.to_csv(file_name, sep=';', encoding='utf-8',index=False)

################################################################################
########################################
#############################################IMPORTS AND LOAD
FILE#####################################################
################################################################################
########################################
def import_and_load_file(file_name):
    import os
    import warnings
    import pandas as pd
    import numpy as np
    warnings.filterwarnings("ignore")

    os.chdir('C:/dersler/capstone')

    dp = lambda dates: pd.datetime.strptime(dates, '%Y-%m-%d')

    capstoneDataSet = pd.read_csv(file_name, sep=';', encoding="ISO-8859-
1",
                                  dtype={'Quarter':
str,"Season":str,"Month":str,"Weekday":str},
                                  parse_dates=['Date'],date_parser=dp)

    return capstoneDataSet

def getList(withoutCurrency):
    list = [
        "CarbonPriceInUSD",
        "CoalPriceInUSD",
        "ElectricityPriceInUSD",
        "NaturalGasPriceInUSD",
        "EuroUSD",
        "GdpUSD",
        "GdpEuro"
    ]
    listWithoutCurrency = [
        "CarbonPriceInUSD",
        "CoalPriceInUSD",
        "ElectricityPriceInUSD",
        "NaturalGasPriceInUSD"
    ]

    if (withoutCurrency):
        return listWithoutCurrency
```

```
    else:
        return list

#########################################################################
########################################
#############################################LOGARITMA#####################
########################################
#########################################################################
########################################

#withoutCurrency = True
file_name = "capstone.csv"
if (withoutCurrency):
    file_name = "capstone_without_cur.csv"

capstoneDataSet = import_and_load_file(file_name)


def to_logaritma(df,column_name):
    df[column_name ] = np.log(df[column_name ])
    df[column_name + "_D1"] = np.log(df[column_name + "_D1"])
    df[column_name + "_D2"] = np.log(df[column_name + "_D2"])
    df[column_name + "_D3"] = np.log(df[column_name + "_D3"])

    df[column_name + "_W1"] = np.log(df[column_name + "_W1"])
    df[column_name + "_AW1"] = np.log(df[column_name + "_AW1"])
    df[column_name + "_W2"] = np.log(df[column_name + "_W2"])
    df[column_name + "_W3"] = np.log(df[column_name + "_W3"])

    df[column_name + "_M1"] = np.log(df[column_name + "_M1"])
    df[column_name + "_AM1"] = np.log(df[column_name + "_AM1"])
    df[column_name + "_M2"] = np.log(df[column_name + "_M2"])
    df[column_name + "_M3"] = np.log(df[column_name + "_M3"])
    df[column_name + "_AM3"] = np.log(df[column_name + "_AM3"])

    return df
#Convert mail varriables to log

list = getList(withoutCurrency)

for column_name in list:
    capstoneDataSet = to_logaritma(capstoneDataSet, column_name)


capstoneDataSet.head(20)

#capstoneDataSet = capstoneDataSet[(capstoneDataSet["Year"] != 2009) ]
capstoneDataSet["Year"].unique()
capstoneDataSet.head(10)


file_name = "capstone_log.csv"
if (withoutCurrency):
    file_name = "capstone_log_without_cur.csv"
capstoneDataSet.to_csv(file_name, sep=';', encoding='utf-8',index=False)

#########################################################################
########################################
#################################################DUMMY
VARIABLE#################################################
#########################################################################
```

```
########################################


#withoutCurrency = True
file_name = "capstone.csv"
if (withoutCurrency):
    file_name = "capstone_without_cur.csv"
capstoneDataSet = import_and_load_file(file_name)

#capstoneDataSet.Month=capstoneDataSet.Month.astype(str)
#capstoneDataSet.Year=capstoneDataSet.Year.astype(str)
#del capstoneDataSet["YearMonth"]        # Remove

def create_dummies( df, colname ):
    col_dummies = pd.get_dummies(df[colname]).rename(columns=lambda x:
colname + '_' + str(x))
    col_dummies.drop(col_dummies.columns[0], axis=1, inplace=True)
    df = pd.concat([df, col_dummies], axis=1)
    df.drop( colname, axis = 1, inplace = True )
    return df

categorical = capstoneDataSet.dtypes[capstoneDataSet.dtypes ==
"object"].index
print(categorical)
#Index(['Month', 'Year', 'Weekday', 'Quarter', 'Season'], dtype='object')
colname = "Month"
#there is no January
capstoneDataSet = create_dummies(capstoneDataSet,colname)
colname = "Weekday"
capstoneDataSet = create_dummies(capstoneDataSet,colname)
colname = "Quarter"
capstoneDataSet = create_dummies(capstoneDataSet,colname)
colname = "Season"
capstoneDataSet = create_dummies(capstoneDataSet,colname)

"""
dummies = pd.get_dummies(capstoneDataSet['Year']).rename(columns=lambda x:
'Year_' + str(x))
capstoneDataSet = pd.concat([capstoneDataSet, dummies], axis=1)
del capstoneDataSet["Year"]        # Remove
"""

file_name = "capstone_cat.csv"
if (withoutCurrency):
    file_name = "capstone_cat_without_cur.csv"
#capstoneDataSet = capstoneDataSet[(capstoneDataSet["Year"] != 2009) ]
capstoneDataSet.to_csv(file_name, sep=';', encoding='utf-8',index=False)

########################################################################
########################################
########################################################################
########################################


file_name = "capstone_log.csv"
if (withoutCurrency):
    file_name = "capstone_log_without_cur.csv"
capstoneDataSet = import_and_load_file(file_name)

categorical = capstoneDataSet.dtypes[capstoneDataSet.dtypes ==
"object"].index
```

```
print(categorical)
#Index(['Month', 'Year', 'Weekday', 'Quarter', 'Season'], dtype='object')
colname = "Month"
#there is no January
capstoneDataSet = create_dummies(capstoneDataSet,colname)
colname = "Weekday"
capstoneDataSet = create_dummies(capstoneDataSet,colname)
colname = "Quarter"
capstoneDataSet = create_dummies(capstoneDataSet,colname)
colname = "Season"
capstoneDataSet = create_dummies(capstoneDataSet,colname)

file_name = "capstone_log_cat.csv"
if (withoutCurrency):
    file_name = "capstone_log_without_cur_cat.csv"
capstoneDataSet.to_csv(file_name, sep=';', encoding='utf-8',index=False)

#########################################################################
########################################
#################################################Correlation
matrix#################################################
#########################################################################
########################################

"""
file_name = "capstone_log_without2009_without_cur_cat.csv"
if (withoutCurrency):
    file_name = "capstone_without_cur.csv"
capstoneDataSet = import_and_load_file(file_name)

import seaborn as sns
import matplotlib.pyplot as plt

corr = capstoneDataSet.corr()
sns.heatmap(corr,
            xticklabels=corr.columns.values,
            yticklabels=corr.columns.values)


fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(corr,cmap='coolwarm', vmin=-1, vmax=1)
fig.colorbar(cax)
ticks = np.arange(0,len(capstoneDataSet.columns),1)
ax.set_xticks(ticks)
plt.xticks(rotation=90)
ax.set_yticks(ticks)
ax.set_xticklabels(capstoneDataSet.columns)
ax.set_yticklabels(capstoneDataSet.columns)
plt.show()

#########################################################################
########################################
#################################################Feature
selection#################################################
#########################################################################
########################################

#########################################################################
########################################
#########################################################################
```

```
########################################
##########################################DATA PREP PART
TWO#########################################################
#############################################################################
########################################


import numpy as np
import os

import warnings

from time import time, sleep  # to keep track of the processing time
import random
from datetime import datetime, timedelta
import pandas as pd

os.chdir('C:/dersler/capstone')
file_name = "capstone_v2.csv"
dp = lambda dates: pd.datetime.strptime(dates, '%Y-%m-%d')

capstoneDataSet = pd.read_csv(file_name, sep=';', encoding="ISO-8859-1",
                              dtype={'Quarter':
str,"Season":str,"Month":str,"Weekday":str},
                              parse_dates=['Date'],date_parser=dp)


capstoneDataSet.dtypes
capstoneDataSet.shape



capstoneDataSet['Weekday'] = capstoneDataSet['Date'].apply(lambda x:
x.strftime('%w'))



capstoneDataSet['Weekday'] = capstoneDataSet['Date'].apply(lambda x:
x.strftime('%a'))

# save to file for later use
file_name = "capstone.csv"
capstoneDataSet.to_csv(file_name, sep=';', encoding='utf-8', index=False)


capstoneDataSet.head(10)

#############################################################################
########################################
#############################################################################
########################################
#############################################################################
########################################
#############################################################################
########################################


import numpy as np
import pandas as pd
import os

import seaborn as sns
```

```
import warnings

from time import time, sleep  # to keep track of the processing time
import random
from sklearn import metrics
from datetime import datetime
import pandas as pd
import matplotlib.pylab as plt
from matplotlib.pylab import rcParams
rcParams['figure.figsize'] = 15, 6
warnings.filterwarnings("ignore")

os.chdir('C:/dersler/capstone')
file_name = "capstone.csv"
dp = lambda dates: pd.datetime.strptime(dates, '%Y-%m-%d')

capstoneDataSet = pd.read_csv(file_name, sep=';', encoding="ISO-8859-1",
                              dtype={'Date': str, 'Quarter':
str,"Season":str,"Month":str},

parse_dates=['Date'],date_parser=dp,index_col="Date")


capstoneDataSet.head()
#now date is the index column

capstoneDataSet.dtypes
capstoneDataSet.shape
capstoneDataSet.index


carbon = capstoneDataSet["CarbonPriceInUSD"]
carbon.columns = ["CarbonPriceInUSD"]
carbon.head()
carbon.dtypes

carbon[:'2009-01-10']

plt.figure(); carbon[:"2015-12-31"].plot(); plt.legend(loc='best')

carbon["2009-09-01"]

carbon["2010-01-02"]
plt.plot(carbon[:"2010-01-01"])

plt.plot(carbon["2010-01-01":"2011-01-01"])

plt.figure(); ts.plot(style='k--', label='Series'); plt.legend()
"""
```

9. APPENDIX B: R SCENARIO 1 CODE

```
#Scenario 1: YEAR=2013 is in test set
library(tree)
library(ISLR)
library(randomForest)

library(MASS)
library(ggplot2)
#install.packages("ISLR")
library(plotROC)
#install.packages("leaps")
library(leaps)
#install.packages("glmnet")
library(glmnet)
library(dplyr)
library(e1071)


setwd("C:/dersler/capstone")
list.files()

file_name <- "capstone_cat_without_cur.csv"

capstoneDataSet<-read.csv(file_name,sep= ";",as.is = TRUE,stringsAsFactors
= F)
capstoneDataSet$Date <- as.Date(capstoneDataSet$Date,format="%Y-%m-%d")


head(capstoneDataSet)
dim(capstoneDataSet)
summary(capstoneDataSet)
str(capstoneDataSet)
names(capstoneDataSet)




#capstoneDataSet$Year<- factor(capstoneDataSet$Year)

ggplot(aes(x=Date, y=CarbonPriceInUSD),data=capstoneDataSet) +
  geom_line(alpha=1/2, color='blue') +
  geom_smooth(color="red",method="auto") +
  geom_smooth(color="green", method="loess") +
  geom_smooth(color="blue", method="lm") +
  ylab(label="Carbon Price in USD") +
  xlab("Date") +
  labs(title = "Plot 1", subtitle = "Carbon Price in Time") +
  scale_y_continuous(limits=c(0,30))

ggplot(aes(x=Date, y=CarbonPriceInUSD_C12),data=capstoneDataSet) +
  geom_point(alpha=1/2) +
  geom_smooth(color="red",method="auto") +
  ylab(label="Carbon Price in USD") +
  xlab("Date") +
  ggtitle("Carbon Price in Time")  + scale_y_continuous(limits=c(-5,5))



ggplot(aes(x=Date, y=CoalPriceInUSD),data=capstoneDataSet) +
```

```
  geom_line(alpha=1/2, color='blue') +
  geom_smooth(color="red",method="auto") +
  geom_smooth(color="green", method="loess") +
  geom_smooth(color="blue", method="lm") +
  ylab(label="Coal Price in USD") +
  xlab("Date") +
  labs(title = "Plot 2", subtitle = "Coal Price in Time") +
  scale_y_continuous(limits=c(0,125))


ggplot(aes(x=Date, y=NaturalGasPriceInUSD),data=capstoneDataSet) +
  geom_line(alpha=1/2, color='blue') +
  geom_smooth(color="red",method="auto") +
  geom_smooth(color="green", method="loess") +
  geom_smooth(color="blue", method="lm") +
  ylab(label="Natural Gas Price in USD") +
  xlab("Date") +
  labs(title = "Plot 3", subtitle = "Natural Gas Price in Time") +
  scale_y_continuous(limits=c(0,125))

ggplot(aes(x=Date, y=ElectricityPriceInUSD),data=capstoneDataSet) +
  geom_line(alpha=1/2, color='blue') +
  geom_smooth(color="red",method="auto") +
  geom_smooth(color="green", method="loess") +
  geom_smooth(color="blue", method="lm") +
  ylab(label="Electricity Price in USD") +
  xlab("Date") +
  labs(title = "Plot 4", subtitle = "Electricity Price in Time") +
  scale_y_continuous(limits=c(0,125))


ggplot(aes(x=Date),data=capstoneDataSet) +
  geom_smooth(aes(y=CarbonPriceInUSD), method="auto",color="green") +
  geom_smooth(aes(y=CoalPriceInUSD),method="auto",color="blue") +
  geom_smooth(aes(y=NaturalGasPriceInUSD,method="auto"),color="red") +
  geom_smooth(aes(y=ElectricityPriceInUSD,method="auto"),color="yellow") +
  labs(title = "Plot 5", subtitle = "Combine all Prices") +
  ylab(label="Price in USD") +
  xlab("Date")  +
  scale_y_continuous(limits=c(0,125))


capstoneDataSet <- capstoneDataSet[capstoneDataSet$Year != 2009,]
numeric_columns<-
names(capstoneDataSet)[which(sapply(capstoneDataSet,is.numeric))]
#LOG(X+1) transform for the variables that have SKENNES over 0.75
set.seed(1)
skewed_feats<-sapply(numeric_columns,function(x)
skewness(capstoneDataSet[[x]],na.rm='TRUE'))
skewed_feats<-skewed_feats[abs(skewed_feats)>0.75]
for(x in names(skewed_feats))
{
  if (is.na(x)) next
  capstoneDataSet[[x]]<-log(capstoneDataSet[[x]]+1)
}


set.seed(1)

cor_mat <- cor(capstoneDataSet[ , numeric_columns])
cor_mat
```

```
cor_price <- apply(capstoneDataSet[,numeric_columns],2,
function(col)cor(col, capstoneDataSet$CarbonPriceInUSD))
cor_price.summary<-
data.frame(index=names(capstoneDataSet[,numeric_columns]),coralated=names(c
apstoneDataSet[,numeric_columns]),price_length=cor_price)[is.na(cor_price),
]
nan_cor <- as.vector(cor_price.summary$coralated)
#class(nan_cor)

nan_cor <- as.character(cor_price.summary[,"coralated"])
class(nan_cor)
#[1] "CarbonPriceInUSD_C12" "CarbonPriceInUSD_C23" "CoalPriceInUSD_C12"
"CoalPriceInUSD_C23"

#delete non correlated columns
capstoneDataSet <- capstoneDataSet[, ! names(capstoneDataSet) %in% nan_cor]

capstoneDataSetTrain <- subset(capstoneDataSet,capstoneDataSet$Year %in%
c(2010,2011,2012,2013,2014))
capstoneDataSetTest <- subset(capstoneDataSet,capstoneDataSet$Year %in%
c(2015))

#unique(capstoneDataSetTrain$Year)


library(leaps)


regfit.fwd=regsubsets(CarbonPriceInUSD~.,data=capstoneDataSetTrain,nvmax=88
,method="forward")
summary(regfit.fwd)
which.max(summary(regfit.fwd)$rsq) #best model with 81 feature
coef(regfit.fwd,81)
#backward
regfit.bwd=regsubsets(CarbonPriceInUSD~.,data=capstoneDataSetTrain,nvmax=88
,method="backward")
summary(regfit.bwd)
which.max(summary(regfit.bwd)$rsq) #best model with 81 feature
coef(regfit.bwd,81)

Sys.setlocale("LC_ALL", "English")


###cross validation for forward and backward####
predict.regsubsets=function(object,newdata,id,...){
  form=as.formula(object$call[[2]])
  mat=model.matrix(form,newdata)
  coefi=coef(object,id=id)
  xvars=names(coefi)
  mat[,xvars]%*%coefi
}

k=10
set.seed(1)
folds=sample(1:k,nrow(capstoneDataSetTrain),replace=TRUE)
cv.errors=matrix(NA,k,88, dimnames=list(NULL, paste(1:88)))
for(j in 1:k){
  set.seed(1)
```

```
best.fit=regsubsets(CarbonPriceInUSD~.,data=capstoneDataSetTrain[folds!=j,]
,nvmax=88,method="forward")
  for(i in 1:length(summary(best.fit)$rsq)){
    #print(paste(j, i, sep="---"))
    pred=predict.regsubsets(best.fit,capstoneDataSetTrain[folds==j,],id=i)
    cv.errors[j,i]=mean( (capstoneDataSetTrain$CarbonPriceInUSD[folds==j]-
pred)^2)
  }
}
mean.cv.errors=apply(cv.errors,2,mean)
mean.cv.errors
which.min(mean.cv.errors) #result=20
mean.cv.errors[20] #0.008386013302
par(mfrow=c(1,1))

reg.best=regsubsets(CarbonPriceInUSD~.,data=capstoneDataSetTrain,
nvmax=88,method="forward")
reg.best.coef<-coef(reg.best,20)
#variables for best model
featureSelected<-names(reg.best.coef[-which(names(reg.best.coef) ==
"(Intercept)")])
featureSelected

df.aree <- as.data.frame(t(mean.cv.errors))
MSE <- sapply(df.aree,  function(col)col)
mse_all<-
data.frame(index=names(df.aree),number_of_attribute=as.numeric(names(df.are
e)),mean.cv.errors=MSE)
#str(mse_all)
#plot(mean.cv.errors,type='b')
ggplot(aes(x=number_of_attribute,y=mean.cv.errors),data=mse_all) +
geom_line(color="green",size=1) +
  ylab(label="Mean Square Error") +
  xlab("Number Of Attributes") +
  labs(title = "Figure 14", subtitle = "Forward Selection")


k=10
set.seed(1)
folds=sample(1:k,nrow(capstoneDataSetTrain),replace=TRUE)
cv.errors=matrix(NA,k,88, dimnames=list(NULL, paste(1:88)))
for(j in 1:k){
  set.seed(1)

best.fit=regsubsets(CarbonPriceInUSD~.,data=capstoneDataSetTrain[folds!=j,]
,nvmax=88,method="backward")
  for(i in 1:length(summary(best.fit)$rsq)){
    #print(paste(j, i, sep="---"))
    pred=predict.regsubsets(best.fit,capstoneDataSetTrain[folds==j,],id=i)
    cv.errors[j,i]=mean( (capstoneDataSetTrain$CarbonPriceInUSD[folds==j]-
pred)^2)
  }
}
mean.cv.errors=apply(cv.errors,2,mean)
mean.cv.errors
which.min(mean.cv.errors) #result=70
mean.cv.errors[70] #0.008673742579

df.aree <- as.data.frame(t(mean.cv.errors))
MSE <- sapply(df.aree,  function(col)col)
```

```
mse_all<-
data.frame(index=names(df.aree),number_of_attribute=as.numeric(names(df.are
e)),mean.cv.errors=MSE)
#str(mse_all)
#plot(mean.cv.errors,type='b')
ggplot(aes(x=number_of_attribute,y=mean.cv.errors),data=mse_all) +
geom_line(color="green",size=1) +
  ylab(label="Mean Square Error") +
  xlab("Number Of Attributes") +
  labs(title = "Figure 15", subtitle = "Bacward Selection")

par(mfrow=c(1,1))
plot(mean.cv.errors,type='b')

reg.best=regsubsets(CarbonPriceInUSD~.,data=capstoneDataSetTrain,
nvmax=88,method="backward")
reg.best.coef<-coef(reg.best,70)
#variables for best model
featureSelected<-names(reg.best.coef[-which(names(reg.best.coef) ==
"(Intercept)")])

reg.best=regsubsets(CarbonPriceInUSD~.,data=capstoneDataSetTrain,
nvmax=88,method="forward")
reg.best.coef<-coef(reg.best,20)
#variables for best model
featureSelected<-names(reg.best.coef[-which(names(reg.best.coef) ==
"(Intercept)")])
featureSelected[21]<-'CarbonPriceInUSD'


capstoneDataSetTrain_Old<-capstoneDataSetTrain
capstoneDataSetTest_Old<-capstoneDataSetTest
capstoneDataSetTrain<-capstoneDataSetTrain_Old[featureSelected]
capstoneDataSetTest<-capstoneDataSetTest_Old[featureSelected]


###ridge and lasso##########
library(glmnet)
grid=10^seq(10,-2,length=100)
set.seed(1)
y=capstoneDataSetTrain$CarbonPriceInUSD

x=model.matrix(CarbonPriceInUSD~.,capstoneDataSetTrain)[,-1]
x.test=model.matrix(CarbonPriceInUSD~.,capstoneDataSetTest)[,-1]
y.test=capstoneDataSetTest$CarbonPriceInUSD


#ridge
set.seed(1)

cv.out=cv.glmnet(x,y,alpha=0) #cv.glmnet default 10-fold cross validation
plot(cv.out)
bestlam=cv.out$lambda.min
bestlam
##best lambda value is 0.04681332
ridge.mod=glmnet(x,y,alpha=0,lambda=grid, thresh=1e-12)
ridge.pred=predict(ridge.mod,s=bestlam,newx=x.test)
mean((ridge.pred-y.test)^2)
##mse=0.002924469
capstoneDataSetTest_Old$Ridge <- ridge.pred
```

```
ggplot(aes(x=Date),data=capstoneDataSetTest_Old) +
  geom_line(aes(y=Ridge,colour="Predicted"),size=1) +
  geom_line(aes(y=CarbonPriceInUSD,colour="Actual"),size=1) +
  geom_line(aes(y=Ridge - CarbonPriceInUSD,colour="Difference"),size=1) +
  ylab(label="Log (USD)") +
  xlab("Date") +
 # labs(title = "Figure 16", subtitle = "Ridge Regression") +
  scale_y_continuous(limits=c(-1,3)) +
  scale_colour_manual("", breaks = c("Predicted", "Actual", "Difference"),
                      values = c("Predicted"="red", "Actual"="green",
"Difference"="blue"))

capstoneDataSetTest_Old$Ridge <- NULL


####lasso######
set.seed(1)
lasso.mod=glmnet(x,y,alpha=1,lambda=grid)
plot(lasso.mod)
cv.out=cv.glmnet(x,y,alpha=1)
plot(cv.out)
bestlam=cv.out$lambda.min
bestlam  #0.0001889862
lasso.pred=predict(lasso.mod,s=bestlam,newx=x.test)
mean((lasso.pred-y.test)^2) #MSE 0.00226574

capstoneDataSetTest_Old$Lasso <- lasso.pred

ggplot(aes(x=Date),data=capstoneDataSetTest_Old) +
  geom_line(aes(y=Lasso,colour="Predicted"),size=1) +
  geom_line(aes(y=CarbonPriceInUSD,colour="Actual"),size=1) +
  geom_line(aes(y=Lasso - CarbonPriceInUSD,colour="Difference"),size=1) +
  ylab(label="Log (USD)") +
  xlab("Date") +
  #labs(title = "Figure 17", subtitle = "Lasso Regression") +
  scale_y_continuous(limits=c(-1,3))  +
  scale_colour_manual("", breaks = c("Predicted", "Actual", "Difference"),
                      values = c("Predicted"="red", "Actual"="green",
"Difference"="blue"))

capstoneDataSetTest_Old$Lasso <- NULL

#######random forest######

library(randomForest)
set.seed(1)
cvcapstoneDataSetTrain<-capstoneDataSetTrain
folds <- cut(seq(1,nrow(capstoneDataSetTrain)),breaks=10,labels=FALSE)
mtry<-c(3,5,10)
ntree<-c(50,100,500)
cv.errors=matrix(NA,3,3, dimnames=list(NULL, paste(1:3)))
for (t in 1:length(ntree)){
  for (m in 1:length(mtry)){
    mse<-rep(NA,10)
    for(i in 1:10){
      set.seed(1)
      testIndexes <- which(folds==i,arr.ind=TRUE)
      testData <- cvcapstoneDataSetTrain[testIndexes, ]
      trainData <- cvcapstoneDataSetTrain[-testIndexes, ]
```

```
model=randomForest(CarbonPriceInUSD~.,data=trainData,mtry=mtry[m],ntree=ntr
ee[t],importance=TRUE)
      yhat = predict(model,newdata=testData)
      mse[i]<-mean((yhat-testData$CarbonPriceInUSD)^2)
    }
    #print(paste(t, m, sep="---"))
    # print(mean(mse))
    cv.errors[t,m]<-mean(mse)
  }
}
cv.errors
which(cv.errors== min(cv.errors), arr.ind = TRUE)
#          1          2          3
# [1,] 0.01489912 0.01185736 0.01189444
# [2,] 0.01424577 0.01192615 0.01151370
# [3,] 0.01374540 0.01185453 0.01103050
cv.errors[3,3] #0.01103050
#ntree=500 and mtry=10 have least MSE=0.01103050



set.seed(1)
model=randomForest(CarbonPriceInUSD~.,data=capstoneDataSetTrain,mtry=10,ntr
ee=500,importance=TRUE)
yhat = predict(model,newdata=capstoneDataSetTest)

y.test <- capstoneDataSetTest$CarbonPriceInUSD
mean((yhat-y.test)^2) #MSE 0.003152316

varImp(model)
varImpPlot(model,type=2)

capstoneDataSetTest_Old$RandomForest <- yhat

ggplot(aes(x=Date),data=capstoneDataSetTest_Old) +
  geom_line(aes(y=RandomForest,colour="Predicted"),size=1) +
  geom_line(aes(y=CarbonPriceInUSD,colour="Actual"),size=1) +
  geom_line(aes(y=RandomForest -
CarbonPriceInUSD,colour="Difference"),size=1) +
  ylab(label="Log (USD)") +
  xlab("Date") +
  #labs(title = "Figure 19", subtitle = "RandomForest Regression") +
  scale_y_continuous(limits=c(-1,3)) +
  scale_colour_manual("", breaks = c("Predicted", "Actual", "Difference"),
                      values = c("Predicted"="red", "Actual"="green",
"Difference"="blue"))

capstoneDataSetTest_Old$RandomForest <- NULL


#######XGBOOST(Extreme Gradient Boosting) ########
library("xgboost")
library("caret")
#install.packages("FactoMineR")
library("FactoMineR")
library("e1071")

library("gbm")

set.seed(1)
```

```
numeric_columns<-
names(capstoneDataSetTrain)[which(sapply(capstoneDataSetTrain,is.numeric))]

trainDescr <- capstoneDataSetTrain[numeric_columns]
trainDescr$CarbonPriceInUSD <- NULL
#trainDescr$Year <- NULL
trainClass <- capstoneDataSetTrain$CarbonPriceInUSD
testDescr <- capstoneDataSetTest[numeric_columns]
testDescr$CarbonPriceInUSD <- NULL
#testDescr$Year <- NULL
testClass  <- capstoneDataSetTest$CarbonPriceInUSD

dtrain <- xgb.DMatrix(data = as.matrix(trainDescr),label =
as.matrix(trainClass))
dtest <- xgb.DMatrix(data =
as.matrix(testDescr),label=as.matrix(testClass))

fitControl <- trainControl(## 2-fold CV
  method = "repeatedcv",
  number = 2,
  repeats = 1)
set.seed(1)
xgbFit <- train(trainDescr, trainClass, method="xgbTree", metric = "RMSE",
                trControl = fitControl, verbose = T,
                tuneLength = 4)

plot(xgbFit)
plot(xgbFit, plotType = "level")
xgbFit$results

best(xgbFit$results, metric="RMSE", maximize=F) #183
tolerance(xgbFit$results, metric="RMSE", maximize=F, tol=2) #55
xgbFit$results[55,]

#     eta max_depth gamma colsample_bytree min_child_weight subsample
nrounds       RMSE  Rsquared        MAE
# 121 0.3         4     0              0.8                1 0.8333333
50 0.08307468 0.9368676 0.05350022
# RMSESD RsquaredSD        MAESD
# 121 0.002391148 0.00358635 0.0005307274

params <- list(booster = "gbtree", objective = "reg:linear",eval_metric =
"rmse",nfold = 10,
               eta=0.3, gamma=0, max_depth=4, min_child_weight=1,
               subsample=0.8333333, colsample_bytree=0.8)
set.seed(1)
xgb1 <- xgb.train (params = params, data = dtrain,nrounds = 50)

xgb.importance <- xgb.importance (feature_names = colnames(dtrain),model =
xgb1)
xgb.plot.importance (importance_matrix = xgb.importance[1:20])
pred <- predict(xgb1,newdata = dtest)
mean((pred-testClass)^2) #mse= 0.2042839

capstoneDataSetTest$XGBoost <- pred

ggplot(aes(x=Date),data=capstoneDataSetTest) +
  geom_line(aes(y=XGBoost),color="red",size=1) +
  geom_line(aes(y=CarbonPriceInUSD),color="green",size=1) +
  geom_line(aes(y=XGBoost - CarbonPriceInUSD),color="blue",size=1) +
  ylab(label="Log (USD)") +
```

```
  xlab("Date") +
  labs(title = "Plot 12", subtitle = "XGBoost Regression") +
  scale_y_continuous(limits=c(-1,3))

capstoneDataSetTest$XGBoost <- NULL


#SVM linear
set.seed(1)
tune.out=tune(svm,CarbonPriceInUSD~.,data=capstoneDataSetTrain,
              kernel="linear",
              ranges=list(cost=c(0.001, 0.01, 0.1, 1,5))) #bu fonksiyon
grid serach yapabiliyor
summary(tune.out)
bestmod=tune.out$best.model
summary(bestmod)
#
# Parameters:
#   SVM-Type:  eps-regression
# SVM-Kernel:  linear
# cost:  5
# gamma:  0.05
# epsilon:  0.1
#
#
# Number of Support Vectors:  524


plot(bestmod, capstoneDataSetTrain)
set.seed(1)
pred=predict(bestmod,capstoneDataSetTest)
testClass  <- capstoneDataSetTest$CarbonPriceInUSD

mean((pred-testClass)^2) #mse= 0.00259102

capstoneDataSetTest_Old$SVM <- pred

ggplot(aes(x=Date),data=capstoneDataSetTest_Old) +
  geom_line(aes(y=SVM,colour="Predicted"),size=1) +
  geom_line(aes(y=CarbonPriceInUSD,colour="Actual"),size=1) +
  geom_line(aes(y=SVM - CarbonPriceInUSD,colour="Difference"),size=1) +
  ylab(label="Log (USD)") +
  xlab("Date") +
  #labs(title = "Figure 20", subtitle = "SVM Regression") +
  scale_y_continuous(limits=c(-1,3))  +
  scale_colour_manual("", breaks = c("Predicted", "Actual", "Difference"),
                      values = c("Predicted"="red", "Actual"="green",
"Difference"="blue"))

capstoneDataSetTest_Old$SVM <- NULL

#SVM radial
set.seed(1)

tune.out=tune(svm,CarbonPriceInUSD~.,data=capstoneDataSetTrain,
              kernel="radial",
              ranges=list(cost=c(0.1,1,2,3,4),gamma=c(0.5,1,2,3,4))) #bu
fonksiyon grid serach yapabiliyor
summary(tune.out)
bestmod=tune.out$best.model
summary(bestmod)
```

```
# Parameters:
#   SVM-Type:  eps-regression
# SVM-Kernel:  radial
# cost:  4
# gamma:  0.5
# epsilon:  0.1
#
#
# Number of Support Vectors:  805
plot(bestmod, capstoneDataSetTrain)
pred=predict(bestmod,capstoneDataSetTest)
mean((pred-testClass)^2) #mse= 0.04221082312


#neural network###############


library(neuralnet)
numeric_columns <- unlist(lapply(capstoneDataSetTrain, is.numeric))
maxs <- apply(capstoneDataSetTrain[numeric_columns], 2, max)
mins <- apply(capstoneDataSetTrain[numeric_columns], 2, min)
set.seed(1)
train_ <- as.data.frame(scale(capstoneDataSetTrain[numeric_columns], center
= mins, scale = maxs - mins))
test.cv <- as.data.frame(scale(capstoneDataSetTest[numeric_columns], center
= mins, scale = maxs - mins))

n <- names(train_)
f <- as.formula(paste("CarbonPriceInUSD ~", paste(n[!n %in%
"CarbonPriceInUSD"], collapse = " + ")))
set.seed(1)
nn <- neuralnet(f,data=train_,hidden=c(8,6,4),linear.output=T)
set.seed(1)
pr.nn <- compute(nn,test.cv[,1:20])
pr.nn_ <- pr.nn$net.result*(max(capstoneDataSetTrain$CarbonPriceInUSD)-
min(capstoneDataSetTrain$CarbonPriceInUSD))+min(capstoneDataSetTrain$Carbon
PriceInUSD)
test.r <-
(test.cv$CarbonPriceInUSD)*(max(capstoneDataSetTrain$CarbonPriceInUSD)-
min(capstoneDataSetTrain$CarbonPriceInUSD))+min(capstoneDataSetTrain$Carbon
PriceInUSD)

plot(nn)


MSE.nn <- sum((test.r - pr.nn_)^2)/nrow(test.cv)
#0.007977745443

capstoneDataSetTest_Old$NN <- pr.nn_

ggplot(aes(x=Date),data=capstoneDataSetTest_Old) +
  geom_line(aes(y=SVM,colour="Predicted"),size=1) +
  geom_line(aes(y=CarbonPriceInUSD,colour="Actual"),size=1) +
  geom_line(aes(y=SVM - CarbonPriceInUSD,colour="Difference"),size=1) +
  ylab(label="Log (USD)") +
  xlab("Date") +
  #labs(title = "Figure 21", subtitle = "ANN Regression") +
  scale_y_continuous(limits=c(-1,3))  +
  scale_colour_manual("", breaks = c("Predicted", "Actual", "Difference"),
```

```
                      values = c("Predicted"="red", "Actual"="green",
"Difference"="blue"))

capstoneDataSetTest_Old$NN <- NULL

ggplot(aes(x=Date),data=capstoneDataSetTest_Old) +
  geom_line(aes(y=NN,colour="ANN"),size=1) +
  geom_line(aes(y=CarbonPriceInUSD,colour="Actual"),size=1) +
  geom_line(aes(y=SVM,colour="SVM"),size=1) +
  geom_line(aes(y=Ridge,colour="RR"),size=1) +
  geom_line(aes(y=Lasso,colour="LL"),size=1) +
  geom_line(aes(y=RandomForest,colour="RF"),size=1) +
  ylab(label="Log (USD)") +
  xlab("Date") +
  # labs(title = "Figure 13", subtitle = "ANN Regression") +
  scale_y_continuous(limits=c(-1,3)) +
  scale_colour_manual("", breaks = c("ANN", "Actual",
"SVM","RR","LL","RF"),
                      values = c("ANN"="red", "Actual"="green",
"SVM"="blue","RR"="black","LL"="yellow","RF"="magenta"))
```

10.APPENDIX C: R SCENARIO 2 CODE

```
#Scenario 2: YEAR=2015 is in test set
library(tree)
library(ISLR)
library(randomForest)

library(MASS)
library(ggplot2)
#install.packages("ISLR")
library(plotROC)
#install.packages("leaps")
library(leaps)
#install.packages("glmnet")
library(glmnet)
library(dplyr)
library(e1071)


setwd("C:/dersler/capstone")
list.files()

file_name <- "capstone_cat_without_cur.csv"

capstoneDataSet<-read.csv(file_name,sep= ";",as.is = TRUE,stringsAsFactors
= F)
capstoneDataSet$Date <- as.Date(capstoneDataSet$Date,format="%Y-%m-%d")


head(capstoneDataSet)
dim(capstoneDataSet)
summary(capstoneDataSet)
str(capstoneDataSet)
names(capstoneDataSet)




#capstoneDataSet$Year<- factor(capstoneDataSet$Year)

ggplot(aes(x=Date, y=CarbonPriceInUSD),data=capstoneDataSet) +
  geom_line(alpha=1/2, color='blue') +
  geom_smooth(color="red",method="auto") +
  geom_smooth(color="green", method="loess") +
  geom_smooth(color="blue", method="lm") +
  ylab(label="Carbon Price in USD") +
  xlab("Date") +
  labs(title = "Plot 1", subtitle = "Carbon Price in Time") +
  scale_y_continuous(limits=c(0,30))

ggplot(aes(x=Date, y=CarbonPriceInUSD_C12),data=capstoneDataSet) +
  geom_point(alpha=1/2) +
  geom_smooth(color="red",method="auto") +
  ylab(label="Carbon Price in USD") +
  xlab("Date") +
  ggtitle("Carbon Price in Time")  + scale_y_continuous(limits=c(-5,5))



ggplot(aes(x=Date, y=CoalPriceInUSD),data=capstoneDataSet) +
```

```
  geom_line(alpha=1/2, color='blue') +
  geom_smooth(color="red",method="auto") +
  geom_smooth(color="green", method="loess") +
  geom_smooth(color="blue", method="lm") +
  ylab(label="Coal Price in USD") +
  xlab("Date") +
  labs(title = "Plot 2", subtitle = "Coal Price in Time") +
  scale_y_continuous(limits=c(0,125))


ggplot(aes(x=Date, y=NaturalGasPriceInUSD),data=capstoneDataSet) +
  geom_line(alpha=1/2, color='blue') +
  geom_smooth(color="red",method="auto") +
  geom_smooth(color="green", method="loess") +
  geom_smooth(color="blue", method="lm") +
  ylab(label="Natural Gas Price in USD") +
  xlab("Date") +
  labs(title = "Plot 3", subtitle = "Natural Gas Price in Time") +
  scale_y_continuous(limits=c(0,125))

ggplot(aes(x=Date, y=ElectricityPriceInUSD),data=capstoneDataSet) +
  geom_line(alpha=1/2, color='blue') +
  geom_smooth(color="red",method="auto") +
  geom_smooth(color="green", method="loess") +
  geom_smooth(color="blue", method="lm") +
  ylab(label="Electricity Price in USD") +
  xlab("Date") +
  labs(title = "Plot 4", subtitle = "Electricity Price in Time") +
  scale_y_continuous(limits=c(0,125))


ggplot(aes(x=Date),data=capstoneDataSet) +
  geom_smooth(aes(y=CarbonPriceInUSD), method="auto",color="green") +
  geom_smooth(aes(y=CoalPriceInUSD),method="auto",color="blue") +
  geom_smooth(aes(y=NaturalGasPriceInUSD,method="auto"),color="red") +
  geom_smooth(aes(y=ElectricityPriceInUSD,method="auto"),color="yellow") +
  labs(title = "Plot 5", subtitle = "Combine all Prices") +
  ylab(label="Price in USD") +
  xlab("Date")  +
  scale_y_continuous(limits=c(0,125))


capstoneDataSet <- capstoneDataSet[capstoneDataSet$Year != 2009,]
numeric_columns<-
names(capstoneDataSet)[which(sapply(capstoneDataSet,is.numeric))]
#LOG(X+1) transform for the variables that have SKENNES over 0.75
set.seed(1)
skewed_feats<-sapply(numeric_columns,function(x)
skewness(capstoneDataSet[[x]],na.rm='TRUE'))
skewed_feats<-skewed_feats[abs(skewed_feats)>0.75]
for(x in names(skewed_feats))
{
  if (is.na(x)) next
  capstoneDataSet[[x]]<-log(capstoneDataSet[[x]]+1)
}


set.seed(1)

cor_mat <- cor(capstoneDataSet[ , numeric_columns])
cor_mat
```

```
cor_price <- apply(capstoneDataSet[,numeric_columns],2,
function(col)cor(col, capstoneDataSet$CarbonPriceInUSD))
cor_price.summary<-
data.frame(index=names(capstoneDataSet[,numeric_columns]),coralated=names(c
apstoneDataSet[,numeric_columns]),price_length=cor_price)[is.na(cor_price),
]
nan_cor <- as.vector(cor_price.summary$coralated)
#class(nan_cor)

nan_cor <- as.character(cor_price.summary[,"coralated"])
class(nan_cor)
#[1] "CarbonPriceInUSD_C12" "CarbonPriceInUSD_C23" "CoalPriceInUSD_C12"
"CoalPriceInUSD_C23"

#delete non correlated columns
capstoneDataSet <- capstoneDataSet[, ! names(capstoneDataSet) %in% nan_cor]

capstoneDataSetTrain <- subset(capstoneDataSet,capstoneDataSet$Year %in%
c(2010,2011,2012,2013,2014))
capstoneDataSetTest <- subset(capstoneDataSet,capstoneDataSet$Year %in%
c(2015))

#unique(capstoneDataSetTrain$Year)


library(leaps)


regfit.fwd=regsubsets(CarbonPriceInUSD~.,data=capstoneDataSetTrain,nvmax=88
,method="forward")
summary(regfit.fwd)
which.max(summary(regfit.fwd)$rsq) #best model with 81 feature
coef(regfit.fwd,81)
#backward
regfit.bwd=regsubsets(CarbonPriceInUSD~.,data=capstoneDataSetTrain,nvmax=88
,method="backward")
summary(regfit.bwd)
which.max(summary(regfit.bwd)$rsq) #best model with 81 feature
coef(regfit.bwd,81)

Sys.setlocale("LC_ALL", "English")


###cross validation for forward and backward####
predict.regsubsets=function(object,newdata,id,...){
  form=as.formula(object$call[[2]])
  mat=model.matrix(form,newdata)
  coefi=coef(object,id=id)
  xvars=names(coefi)
  mat[,xvars]%*%coefi
}

k=10
set.seed(1)
folds=sample(1:k,nrow(capstoneDataSetTrain),replace=TRUE)
cv.errors=matrix(NA,k,88, dimnames=list(NULL, paste(1:88)))
for(j in 1:k){
  set.seed(1)
```

```
best.fit=regsubsets(CarbonPriceInUSD~.,data=capstoneDataSetTrain[folds!=j,]
,nvmax=88,method="forward")
  for(i in 1:length(summary(best.fit)$rsq)){
    #print(paste(j, i, sep="---"))
    pred=predict.regsubsets(best.fit,capstoneDataSetTrain[folds==j,],id=i)
    cv.errors[j,i]=mean( (capstoneDataSetTrain$CarbonPriceInUSD[folds==j]-
pred)^2)
  }
}
mean.cv.errors=apply(cv.errors,2,mean)
mean.cv.errors
which.min(mean.cv.errors) #result=20
mean.cv.errors[20] #0.008386013302
par(mfrow=c(1,1))

reg.best=regsubsets(CarbonPriceInUSD~.,data=capstoneDataSetTrain,
nvmax=88,method="forward")
reg.best.coef<-coef(reg.best,20)
#variables for best model
featureSelected<-names(reg.best.coef[-which(names(reg.best.coef) ==
"(Intercept)")])
featureSelected

df.aree <- as.data.frame(t(mean.cv.errors))
MSE <- sapply(df.aree,  function(col)col)
mse_all<-
data.frame(index=names(df.aree),number_of_attribute=as.numeric(names(df.are
e)),mean.cv.errors=MSE)
#str(mse_all)
#plot(mean.cv.errors,type='b')
ggplot(aes(x=number_of_attribute,y=mean.cv.errors),data=mse_all) +
geom_line(color="green",size=1) +
  ylab(label="Mean Square Error") +
  xlab("Number Of Attributes") +
  labs(title = "Figure 14", subtitle = "Forward Selection")


k=10
set.seed(1)
folds=sample(1:k,nrow(capstoneDataSetTrain),replace=TRUE)
cv.errors=matrix(NA,k,88, dimnames=list(NULL, paste(1:88)))
for(j in 1:k){
  set.seed(1)

best.fit=regsubsets(CarbonPriceInUSD~.,data=capstoneDataSetTrain[folds!=j,]
,nvmax=88,method="backward")
  for(i in 1:length(summary(best.fit)$rsq)){
    #print(paste(j, i, sep="---"))
    pred=predict.regsubsets(best.fit,capstoneDataSetTrain[folds==j,],id=i)
    cv.errors[j,i]=mean( (capstoneDataSetTrain$CarbonPriceInUSD[folds==j]-
pred)^2)
  }
}
mean.cv.errors=apply(cv.errors,2,mean)
mean.cv.errors
which.min(mean.cv.errors) #result=70
mean.cv.errors[70] #0.008673742579

df.aree <- as.data.frame(t(mean.cv.errors))
MSE <- sapply(df.aree,  function(col)col)
```

```
mse_all<-
data.frame(index=names(df.aree),number_of_attribute=as.numeric(names(df.are
e)),mean.cv.errors=MSE)
#str(mse_all)
#plot(mean.cv.errors,type='b')
ggplot(aes(x=number_of_attribute,y=mean.cv.errors),data=mse_all) +
geom_line(color="green",size=1) +
  ylab(label="Mean Square Error") +
  xlab("Number Of Attributes") +
  labs(title = "Figure 15", subtitle = "Bacward Selection")

par(mfrow=c(1,1))
plot(mean.cv.errors,type='b')

reg.best=regsubsets(CarbonPriceInUSD~.,data=capstoneDataSetTrain,
nvmax=88,method="backward")
reg.best.coef<-coef(reg.best,70)
#variables for best model
featureSelected<-names(reg.best.coef[-which(names(reg.best.coef) ==
"(Intercept)")])

reg.best=regsubsets(CarbonPriceInUSD~.,data=capstoneDataSetTrain,
nvmax=88,method="forward")
reg.best.coef<-coef(reg.best,20)
#variables for best model
featureSelected<-names(reg.best.coef[-which(names(reg.best.coef) ==
"(Intercept)")])
featureSelected[21]<-'CarbonPriceInUSD'


capstoneDataSetTrain_Old<-capstoneDataSetTrain
capstoneDataSetTest_Old<-capstoneDataSetTest
capstoneDataSetTrain<-capstoneDataSetTrain_Old[featureSelected]
capstoneDataSetTest<-capstoneDataSetTest_Old[featureSelected]


###ridge and lasso##########
library(glmnet)
grid=10^seq(10,-2,length=100)
set.seed(1)
y=capstoneDataSetTrain$CarbonPriceInUSD

x=model.matrix(CarbonPriceInUSD~.,capstoneDataSetTrain)[,-1]
x.test=model.matrix(CarbonPriceInUSD~.,capstoneDataSetTest)[,-1]
y.test=capstoneDataSetTest$CarbonPriceInUSD


#ridge
set.seed(1)

cv.out=cv.glmnet(x,y,alpha=0) #cv.glmnet default 10-fold cross validation
plot(cv.out)
bestlam=cv.out$lambda.min
bestlam
##best lambda value is 0.04681332
ridge.mod=glmnet(x,y,alpha=0,lambda=grid, thresh=1e-12)
ridge.pred=predict(ridge.mod,s=bestlam,newx=x.test)
mean((ridge.pred-y.test)^2)
##mse=0.002924469
capstoneDataSetTest_Old$Ridge <- ridge.pred
```

```
ggplot(aes(x=Date),data=capstoneDataSetTest_Old) +
  geom_line(aes(y=Ridge,colour="Predicted"),size=1) +
  geom_line(aes(y=CarbonPriceInUSD,colour="Actual"),size=1) +
  geom_line(aes(y=Ridge - CarbonPriceInUSD,colour="Difference"),size=1) +
  ylab(label="Log (USD)") +
  xlab("Date") +
 # labs(title = "Figure 16", subtitle = "Ridge Regression") +
  scale_y_continuous(limits=c(-1,3)) +
  scale_colour_manual("", breaks = c("Predicted", "Actual", "Difference"),
                      values = c("Predicted"="red", "Actual"="green",
"Difference"="blue"))

capstoneDataSetTest_Old$Ridge <- NULL


####lasso######
set.seed(1)
lasso.mod=glmnet(x,y,alpha=1,lambda=grid)
plot(lasso.mod)
cv.out=cv.glmnet(x,y,alpha=1)
plot(cv.out)
bestlam=cv.out$lambda.min
bestlam  #0.0001889862
lasso.pred=predict(lasso.mod,s=bestlam,newx=x.test)
mean((lasso.pred-y.test)^2) #MSE 0.00226574

capstoneDataSetTest_Old$Lasso <- lasso.pred

ggplot(aes(x=Date),data=capstoneDataSetTest_Old) +
  geom_line(aes(y=Lasso,colour="Predicted"),size=1) +
  geom_line(aes(y=CarbonPriceInUSD,colour="Actual"),size=1) +
  geom_line(aes(y=Lasso - CarbonPriceInUSD,colour="Difference"),size=1) +
  ylab(label="Log (USD)") +
  xlab("Date") +
  #labs(title = "Figure 17", subtitle = "Lasso Regression") +
  scale_y_continuous(limits=c(-1,3))  +
  scale_colour_manual("", breaks = c("Predicted", "Actual", "Difference"),
                      values = c("Predicted"="red", "Actual"="green",
"Difference"="blue"))

capstoneDataSetTest_Old$Lasso <- NULL

#######random forest######

library(randomForest)
set.seed(1)
cvcapstoneDataSetTrain<-capstoneDataSetTrain
folds <- cut(seq(1,nrow(capstoneDataSetTrain)),breaks=10,labels=FALSE)
mtry<-c(3,5,10)
ntree<-c(50,100,500)
cv.errors=matrix(NA,3,3, dimnames=list(NULL, paste(1:3)))
for (t in 1:length(ntree)){
  for (m in 1:length(mtry)){
    mse<-rep(NA,10)
    for(i in 1:10){
      set.seed(1)
      testIndexes <- which(folds==i,arr.ind=TRUE)
      testData <- cvcapstoneDataSetTrain[testIndexes, ]
      trainData <- cvcapstoneDataSetTrain[-testIndexes, ]
```

```
model=randomForest(CarbonPriceInUSD~.,data=trainData,mtry=mtry[m],ntree=ntr
ee[t],importance=TRUE)
      yhat = predict(model,newdata=testData)
      mse[i]<-mean((yhat-testData$CarbonPriceInUSD)^2)
    }
    #print(paste(t, m, sep="---"))
    # print(mean(mse))
    cv.errors[t,m]<-mean(mse)
  }
}
cv.errors
which(cv.errors== min(cv.errors), arr.ind = TRUE)
#          1          2          3
# [1,] 0.01489912 0.01185736 0.01189444
# [2,] 0.01424577 0.01192615 0.01151370
# [3,] 0.01374540 0.01185453 0.01103050
cv.errors[3,3] #0.01103050
#ntree=500 and mtry=10 have least MSE=0.01103050



set.seed(1)
model=randomForest(CarbonPriceInUSD~.,data=capstoneDataSetTrain,mtry=10,ntr
ee=500,importance=TRUE)
yhat = predict(model,newdata=capstoneDataSetTest)

y.test <- capstoneDataSetTest$CarbonPriceInUSD
mean((yhat-y.test)^2) #MSE 0.003152316

varImp(model)
varImpPlot(model,type=2)

capstoneDataSetTest_Old$RandomForest <- yhat

ggplot(aes(x=Date),data=capstoneDataSetTest_Old) +
  geom_line(aes(y=RandomForest,colour="Predicted"),size=1) +
  geom_line(aes(y=CarbonPriceInUSD,colour="Actual"),size=1) +
  geom_line(aes(y=RandomForest -
CarbonPriceInUSD,colour="Difference"),size=1) +
  ylab(label="Log (USD)") +
  xlab("Date") +
  #labs(title = "Figure 19", subtitle = "RandomForest Regression") +
  scale_y_continuous(limits=c(-1,3)) +
  scale_colour_manual("", breaks = c("Predicted", "Actual", "Difference"),
                      values = c("Predicted"="red", "Actual"="green",
"Difference"="blue"))

capstoneDataSetTest_Old$RandomForest <- NULL


#######XGBOOST(Extreme Gradient Boosting) ########
library("xgboost")
library("caret")
#install.packages("FactoMineR")
library("FactoMineR")
library("e1071")

library("gbm")

set.seed(1)
```

```
numeric_columns<-
names(capstoneDataSetTrain)[which(sapply(capstoneDataSetTrain,is.numeric))]

trainDescr <- capstoneDataSetTrain[numeric_columns]
trainDescr$CarbonPriceInUSD <- NULL
#trainDescr$Year <- NULL
trainClass <- capstoneDataSetTrain$CarbonPriceInUSD
testDescr <- capstoneDataSetTest[numeric_columns]
testDescr$CarbonPriceInUSD <- NULL
#testDescr$Year <- NULL
testClass  <- capstoneDataSetTest$CarbonPriceInUSD

dtrain <- xgb.DMatrix(data = as.matrix(trainDescr),label =
as.matrix(trainClass))
dtest <- xgb.DMatrix(data =
as.matrix(testDescr),label=as.matrix(testClass))

fitControl <- trainControl(## 2-fold CV
  method = "repeatedcv",
  number = 2,
  repeats = 1)
set.seed(1)
xgbFit <- train(trainDescr, trainClass, method="xgbTree", metric = "RMSE",
                trControl = fitControl, verbose = T,
                tuneLength = 4)

plot(xgbFit)
plot(xgbFit, plotType = "level")
xgbFit$results

best(xgbFit$results, metric="RMSE", maximize=F) #183
tolerance(xgbFit$results, metric="RMSE", maximize=F, tol=2) #55
xgbFit$results[55,]

#     eta max_depth gamma colsample_bytree min_child_weight subsample
nrounds       RMSE  Rsquared        MAE
# 121 0.3         4     0              0.8                1 0.8333333
50 0.08307468 0.9368676 0.05350022
# RMSESD RsquaredSD        MAESD
# 121 0.002391148 0.00358635 0.0005307274

params <- list(booster = "gbtree", objective = "reg:linear",eval_metric =
"rmse",nfold = 10,
               eta=0.3, gamma=0, max_depth=4, min_child_weight=1,
               subsample=0.8333333, colsample_bytree=0.8)
set.seed(1)
xgb1 <- xgb.train (params = params, data = dtrain,nrounds = 50)

xgb.importance <- xgb.importance (feature_names = colnames(dtrain),model =
xgb1)
xgb.plot.importance (importance_matrix = xgb.importance[1:20])
pred <- predict(xgb1,newdata = dtest)
mean((pred-testClass)^2) #mse= 0.2042839

capstoneDataSetTest$XGBoost <- pred

ggplot(aes(x=Date),data=capstoneDataSetTest) +
  geom_line(aes(y=XGBoost),color="red",size=1) +
  geom_line(aes(y=CarbonPriceInUSD),color="green",size=1) +
  geom_line(aes(y=XGBoost - CarbonPriceInUSD),color="blue",size=1) +
  ylab(label="Log (USD)") +
```

```
  xlab("Date") +
  labs(title = "Plot 12", subtitle = "XGBoost Regression") +
  scale_y_continuous(limits=c(-1,3))

capstoneDataSetTest$XGBoost <- NULL


#SVM linear
set.seed(1)
tune.out=tune(svm,CarbonPriceInUSD~.,data=capstoneDataSetTrain,
              kernel="linear",
              ranges=list(cost=c(0.001, 0.01, 0.1, 1,5))) #bu fonksiyon
grid serach yapabiliyor
summary(tune.out)
bestmod=tune.out$best.model
summary(bestmod)
#
# Parameters:
#   SVM-Type:  eps-regression
# SVM-Kernel:  linear
# cost:  5
# gamma:  0.05
# epsilon:  0.1
#
#
# Number of Support Vectors:  524


plot(bestmod, capstoneDataSetTrain)
set.seed(1)
pred=predict(bestmod,capstoneDataSetTest)
testClass  <- capstoneDataSetTest$CarbonPriceInUSD

mean((pred-testClass)^2) #mse= 0.00259102

capstoneDataSetTest_Old$SVM <- pred

ggplot(aes(x=Date),data=capstoneDataSetTest_Old) +
  geom_line(aes(y=SVM,colour="Predicted"),size=1) +
  geom_line(aes(y=CarbonPriceInUSD,colour="Actual"),size=1) +
  geom_line(aes(y=SVM - CarbonPriceInUSD,colour="Difference"),size=1) +
  ylab(label="Log (USD)") +
  xlab("Date") +
  #labs(title = "Figure 20", subtitle = "SVM Regression") +
  scale_y_continuous(limits=c(-1,3))  +
  scale_colour_manual("", breaks = c("Predicted", "Actual", "Difference"),
                      values = c("Predicted"="red", "Actual"="green",
"Difference"="blue"))

capstoneDataSetTest_Old$SVM <- NULL

#SVM radial
set.seed(1)

tune.out=tune(svm,CarbonPriceInUSD~.,data=capstoneDataSetTrain,
              kernel="radial",
              ranges=list(cost=c(0.1,1,2,3,4),gamma=c(0.5,1,2,3,4))) #bu
fonksiyon grid serach yapabiliyor
summary(tune.out)
bestmod=tune.out$best.model
summary(bestmod)
```

```
# Parameters:
#   SVM-Type:  eps-regression
# SVM-Kernel:  radial
# cost:  4
# gamma:  0.5
# epsilon:  0.1
#
#
# Number of Support Vectors:  805
plot(bestmod, capstoneDataSetTrain)
pred=predict(bestmod,capstoneDataSetTest)
mean((pred-testClass)^2) #mse= 0.04221082312


#neural network###############


library(neuralnet)
numeric_columns <- unlist(lapply(capstoneDataSetTrain, is.numeric))
maxs <- apply(capstoneDataSetTrain[numeric_columns], 2, max)
mins <- apply(capstoneDataSetTrain[numeric_columns], 2, min)
set.seed(1)
train_ <- as.data.frame(scale(capstoneDataSetTrain[numeric_columns], center
= mins, scale = maxs - mins))
test.cv <- as.data.frame(scale(capstoneDataSetTest[numeric_columns], center
= mins, scale = maxs - mins))

n <- names(train_)
f <- as.formula(paste("CarbonPriceInUSD ~", paste(n[!n %in%
"CarbonPriceInUSD"], collapse = " + ")))
set.seed(1)
nn <- neuralnet(f,data=train_,hidden=c(8,6,4),linear.output=T)
set.seed(1)
pr.nn <- compute(nn,test.cv[,1:20])
pr.nn_ <- pr.nn$net.result*(max(capstoneDataSetTrain$CarbonPriceInUSD)-
min(capstoneDataSetTrain$CarbonPriceInUSD))+min(capstoneDataSetTrain$Carbon
PriceInUSD)
test.r <-
(test.cv$CarbonPriceInUSD)*(max(capstoneDataSetTrain$CarbonPriceInUSD)-
min(capstoneDataSetTrain$CarbonPriceInUSD))+min(capstoneDataSetTrain$Carbon
PriceInUSD)

plot(nn)


MSE.nn <- sum((test.r - pr.nn_)^2)/nrow(test.cv)
#0.007977745443

capstoneDataSetTest_Old$NN <- pr.nn_

ggplot(aes(x=Date),data=capstoneDataSetTest_Old) +
  geom_line(aes(y=SVM,colour="Predicted"),size=1) +
  geom_line(aes(y=CarbonPriceInUSD,colour="Actual"),size=1) +
  geom_line(aes(y=SVM - CarbonPriceInUSD,colour="Difference"),size=1) +
  ylab(label="Log (USD)") +
  xlab("Date") +
  #labs(title = "Figure 21", subtitle = "ANN Regression") +
  scale_y_continuous(limits=c(-1,3))  +
  scale_colour_manual("", breaks = c("Predicted", "Actual", "Difference"),
```

```
                      values = c("Predicted"="red", "Actual"="green",
"Difference"="blue"))

capstoneDataSetTest_Old$NN <- NULL

ggplot(aes(x=Date),data=capstoneDataSetTest_Old) +
  geom_line(aes(y=NN,colour="ANN"),size=1) +
  geom_line(aes(y=CarbonPriceInUSD,colour="Actual"),size=1) +
  geom_line(aes(y=SVM,colour="SVM"),size=1) +
  geom_line(aes(y=Ridge,colour="RR"),size=1) +
  geom_line(aes(y=Lasso,colour="LL"),size=1) +
  geom_line(aes(y=RandomForest,colour="RF"),size=1) +
  ylab(label="Log (USD)") +
  xlab("Date") +
  # labs(title = "Figure 13", subtitle = "ANN Regression") +
  scale_y_continuous(limits=c(-1,3)) +
  scale_colour_manual("", breaks = c("ANN", "Actual",
"SVM","RR","LL","RF"),
                      values = c("ANN"="red", "Actual"="green",
"SVM"="blue","RR"="black","LL"="yellow","RF"="magenta"))
```

11.REFERENCES

1. B. Zhu, D. Han, P. Wang “Forecasting Carbon prices using empirical mode decomposition and evolutionary least squares support vector regression” 521-530 (2017)
2. G. James, D. Witten, T. Hastie, R. Tibshirani “An Introduction to Statistical Learning” 337-268 (2013)
3. E. Alpaydın “Introduction to Machine Learning” 218-225 (2004)
4. G. James, D. Witten, T. Hastie, R. Tibshirani “An Introduction to Statistical Learning” 316-320 (2013)
5. J. Mei, D. He, R.G. Harley “A random forest method for real-time price forecasting in New York Electricity Market” 3-4 (2014)
6. B. Zhu “A Novel Multiscale Ensemble Carbon Price Prediction Model Integrating Empirical Mode Decomposition, Genetic Algorithm and Artificial Neural Network” 3-5 (2012)
7. M. Tsai, Y. Kuo “A Forecasting System of Carbon Price in the Carbon Trading Markets Using Artificial Neural Network” 2-5 (2013)
8. E. Alpaydın “Introduction to Machine Learning” 229-267 (2004)
9. Z. Yang, L Ce, L. Lian “Electricity price forecasting by a hybrid model, combining wavelet transform, ARMA and kernel-based extreme learning machine methods” 2-3 (2017)
10. G. James, D. Witten, T. Hastie, R. Tibshirani “An Introduction to Statistical Learning” 215-219 (2013)
11. G. James, D. Witten, T. Hastie, R. Tibshirani “An Introduction to Statistical Learning” 219-227 (2013)
12. "The Grammar of Graphics" https://cran.r-project.org/web/packages/ggplot2/index.html
13. European Climate Exchange. http://www.europeanclimateexchange.com/
14. D. Keles, J. Scelle, F. Paraschiv , W. Fichtner “Extended forecast methods for day-ahead electricity spot prices applying artificial neural networks” 218-230 (2015)
15. G. James, D. Witten, T. Hastie, R. Tibshirani “An Introduction to Statistical Learning” 86-88 (2013)
16. G. James, D. Witten, T. Hastie, R. Tibshirani “An Introduction to Statistical Learning” 247 (2013)
17. G. James, D. Witten, T. Hastie, R. Tibshirani “An Introduction to Statistical Learning” 248 (2013)

18. “Smoothed Conditional Mean” https://www.rdocumentation.org/packages/ggplot2/versions/1.0.1/topics/geom_smooth
19. “skewness” https://www.rdocumentation.org/packages/e1071/versions/1.7-0/topics/skewness
20. “Logarithms and Exponentials” https://www.rdocumentation.org/packages/base/versions/3.5.1/topics/log
21. G. James, D. Witten, T. Hastie, R. Tibshirani “An Introduction to Statistical Learning” 176-184 (2013)
22. “Glmnet” https://www.rdocumentation.org/packages/glmnet/versions/2.0-16/topics/glmnet
23. G. James, D. Witten, T. Hastie, R. Tibshirani “An Introduction to Statistical Learning” 227-228 (2013)
24. “tune” https://www.rdocumentation.org/packages/e1071/versions/1.7-0/topics/tune
25. G. James, D. Witten, T. Hastie, R. Tibshirani “An Introduction to Statistical Learning” 215 (2013)
26. “Xgboost” https://www.rdocumentation.org/packages/xgboost/versions/0.4-4/topics/xgboost