IS624 - Final

James Quacinella

07/12/2015

Contents

Introduction

Originally, I planned on working with a few data sets relating to electrical power generators and their $C0_2$ emissions. I found a few data sets, but a critical data set on world-wide generators is not free sadly. In lieu of working with this data, I found a dataset on the UCI Machine Learning repository that was tangentially related to the original idea. To quote directly from the data source itself:

> The dataset contains 9568 data points collected from a Combined Cycle Power Plant over 6 years (2006-2011), when the power plant was set to work with full load. Features consist of hourly average ambient variables Temperature (T), Ambient Pressure (AP), Relative Humidity (RH) and Exhaust Vacuum (V) to predict the net hourly electrical energy output (EP) of the plant.
>
> A combined cycle power plant (CCPP) is composed of gas turbines (GT), steam turbines (ST) and heat recovery steam generators. In a CCPP, the electricity is generated by gas and steam turbines, which are combined in one cycle, and is transferred from one turbine to another. While the Vacuum is collected from and has an effect on the Steam Turbine, the other three o the ambient variables effect the GT performance.

In this project, I will try to develop predictive models for this dataset. Here, I will try to predict the energy output (EP) of the power plant from the 4 predictors gievn (temperature, ambient pressure, relative humidity and exhaust vaccum). My hypothesis is that models regarding physical processes should be pretty accurate once a proper model is chosen, since physical processes obey physical laws.

I will start with linear regression and see what other models can bring.

Data Preperation

The first thing to do was convert the .ods file into a .csv file using LibreOffice. The data.csv file is the result, and has been loaded:

```
# Init
library(e1071)
library(caret)
library(corrplot)
library(ggplot2)
library(GGally)
library(forecast)

# Set random set for predictability
set.seed(200)

# Load data
df <- read.csv("CCPP/data.csv")
predictors <-  c("V", "AT", "RH", "AP")
#df.predictors <- df[ , c("V", "AT", "RH", "AP")]
#df.predict <- data.frame(PE=df[ , c("PE")])
```

Lets see if there are any non-complete cases in the data:

```
sum(complete.cases(df$AT)) == nrow(df) # True
sum(complete.cases(df$V)) == nrow(df)  # True
sum(complete.cases(df$AP)) == nrow(df) # True
sum(complete.cases(df$RH)) == nrow(df) # True
```

It looks like we have a full data set with no missing NA values. Lets see if any of the predictors are significantly skewed:

```
skewValues <- apply(df[, predictors], 2, skewness)
head(skewValues)
```

```
##          V         AT         RH         AP
##  0.1984588 -0.1363503 -0.4317034  0.2653615
```

From this I would say that there is no significant skewness here, so Box Cox transformations are not needed.

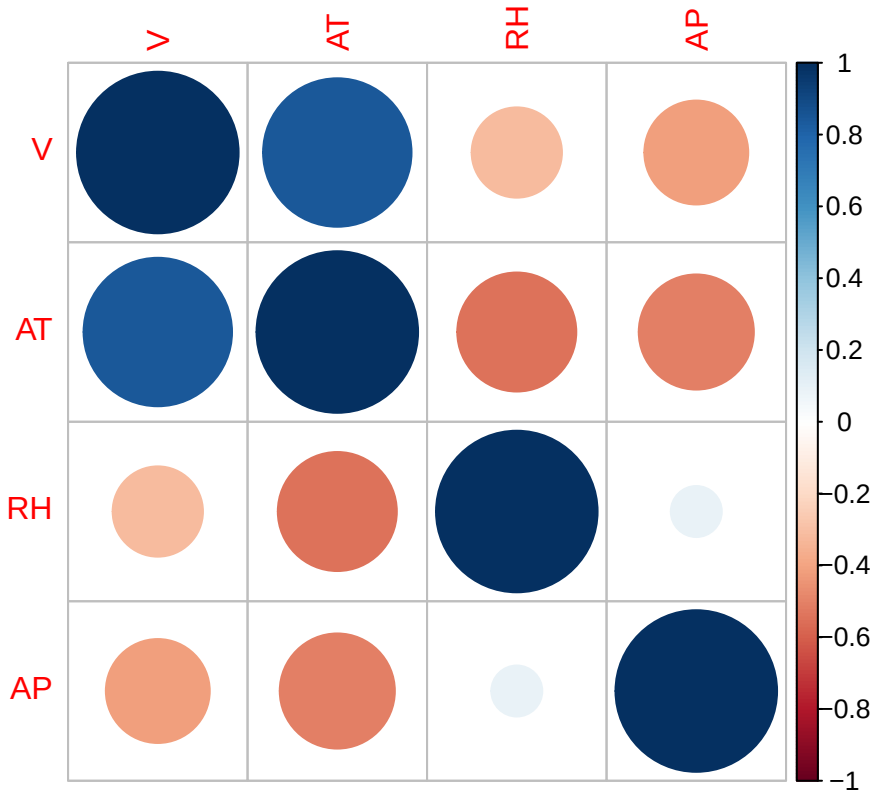
TODO: * should we transform via center and scaling?

Lets remove any near zero variance predictors:

```
remove <- nearZeroVar(df[ , predictors])   # No columns are near zero variance
```

Are there any correlations between predictors?

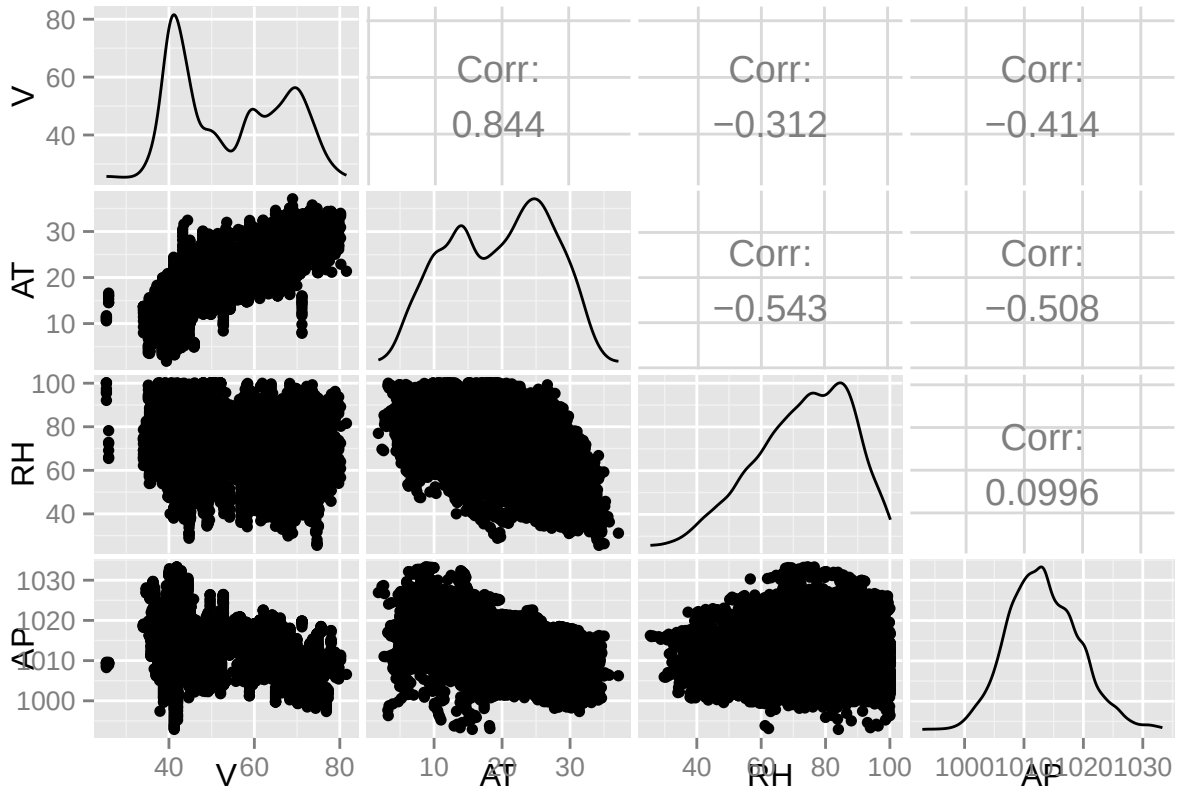
```
correlations <- cor(df[ , predictors])
corrplot::corrplot(correlations, order = "hclust")
```



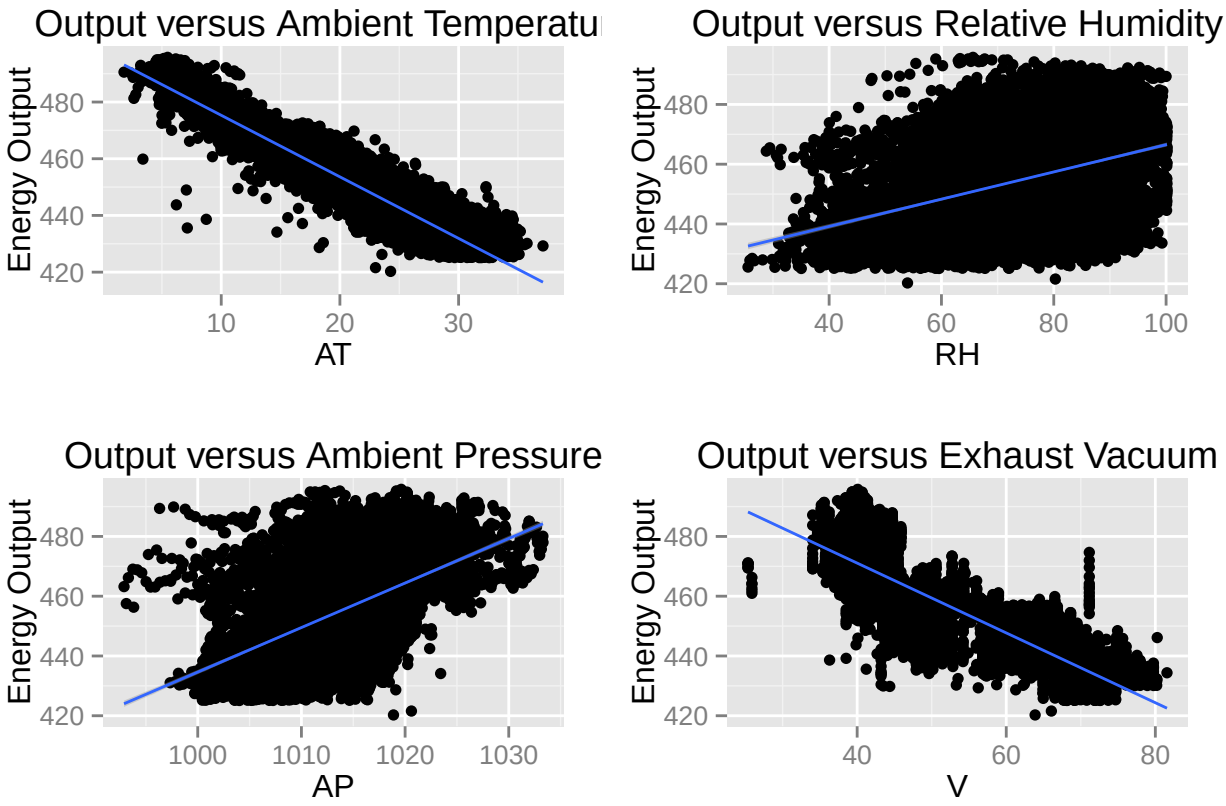
From this we can see some correlations between predictors, but that might be normal: the operating conditions of the plant probably change according to fundamental physical laws and operating contraints, so their changes over time might be correlated.

Lets view that data:

```
ggpairs(df[ , predictors])
```




```
p1 <- ggplot(df, aes(x=AT, y=PE)) + geom_point() + geom_smooth(method='lm') + ggtitle("Output versus Amb
p2 <- ggplot(df, aes(x=AP, y=PE)) + geom_point() + geom_smooth(method='lm') + ggtitle("Output versus Amb
p3 <- ggplot(df, aes(x=RH, y=PE)) + geom_point() + geom_smooth(method='lm') + ggtitle("Output versus Rel
p4 <- ggplot(df, aes(x=V, y=PE)) + geom_point() + geom_smooth(method='lm') + ggtitle("Output versus Exha
multiplot(p1, p2, p3, p4, cols=2)
```



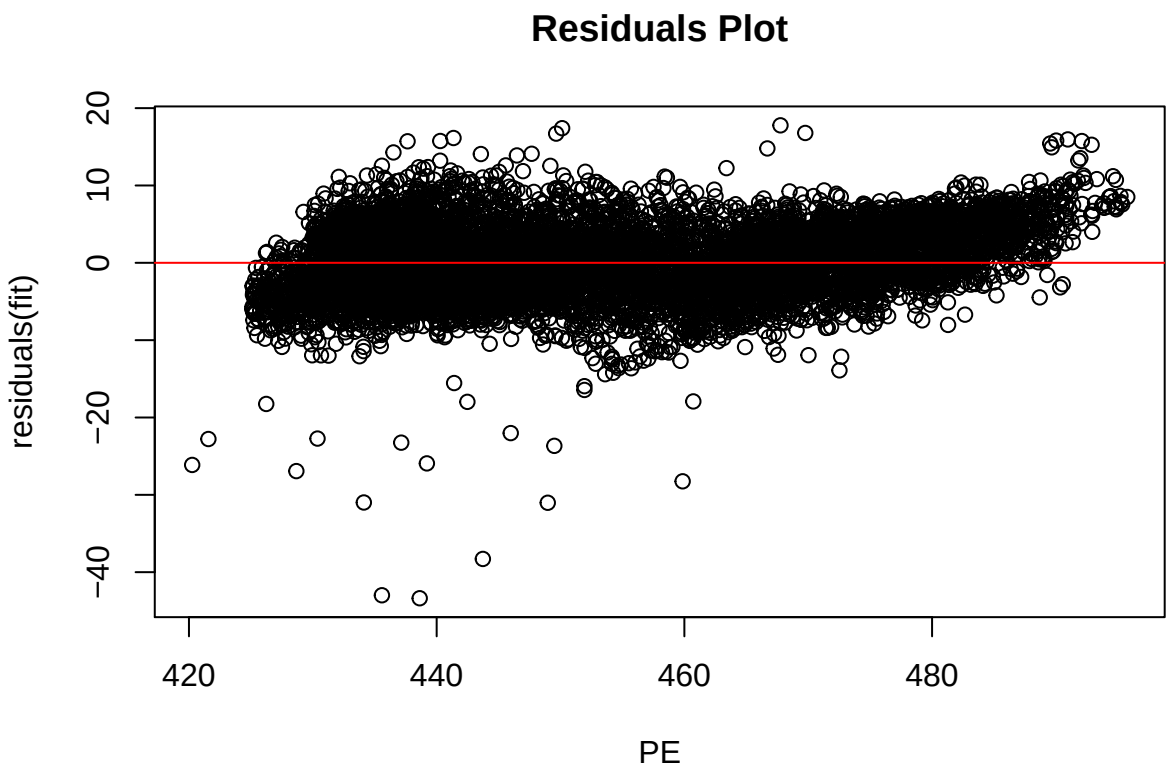
For the next steps, where we build models for this data, lets create a training and test set for performance measures. I played around with the percentage of data to be used for cross-validation, with 90% used for training being a good value that ends with good results (though the range of values tried always ended up with results with $R^2 > .9$:

```
trainIndex <- createDataPartition(df$PE, p = .9, list=FALSE)
df.training <- df[trainIndex, ]
df.test <- df[-trainIndex, ]
```

Linear Regression Model

```
# Build Lineat Regression Model
fit  <- lm(PE ~ V + AT + RH + AP, data=df.training)
#summary(fit)

# Residuals
plot(residuals(fit) ~ PE, data=df.training, main="Residuals Plot")
abline(0, 0, col='red')
```



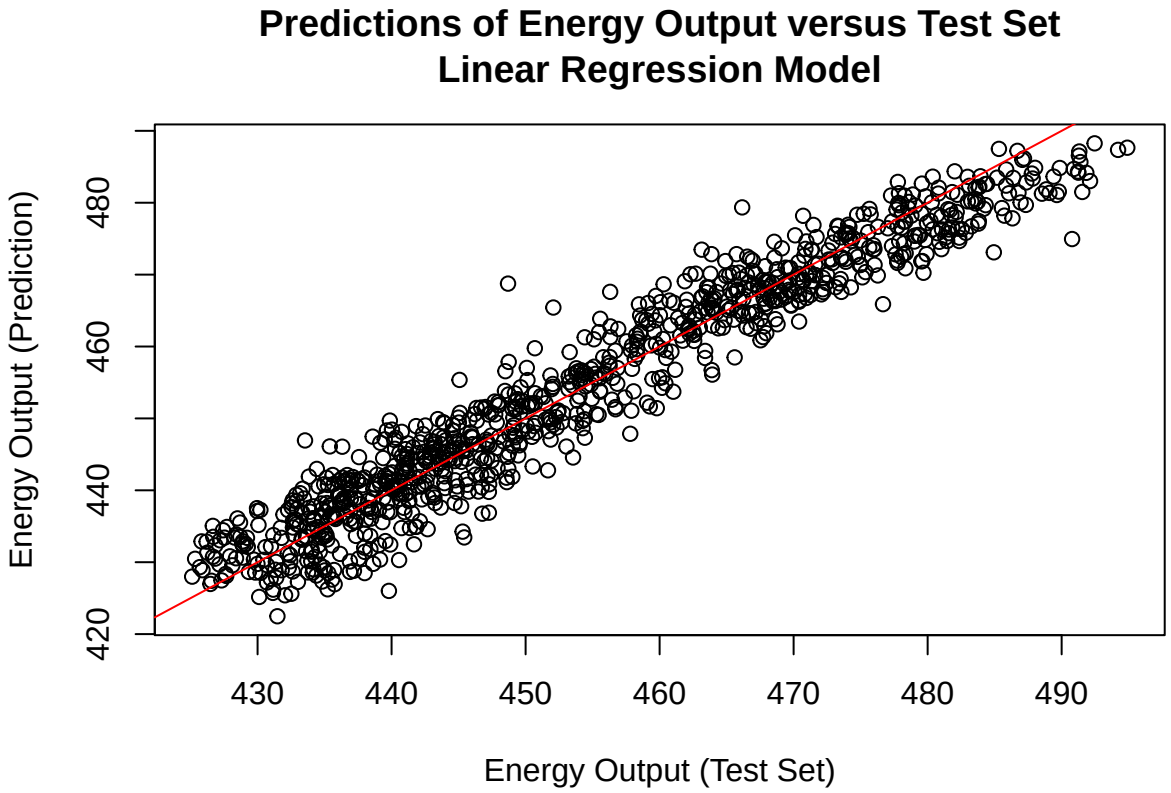

```
# Forecast the data
forecast <- forecast(fit, newdata=df.test)
accuracy(forecast, df.test)
```

```
##                         ME     RMSE      MAE          MPE      MAPE
## Training set -6.314173e-16 4.577031 3.638568 -0.009876987 0.8024418
## Test set      1.732372e-01 4.374684 3.514657  0.027573692 0.7745421
##                   MASE
## Training set 0.2455488
## Test set     0.2371866
```

```
postResample(pred = forecast$mean, obs = df.test$PE)
```

```
##      RMSE  Rsquared
## 4.3746838 0.9363484
```

```
# Plot predictions versus reality in test set
plot(df.test$PE, forecast$mean, main="Predictions of Energy Output versus Test Set\nLinear Regression M
abline(0, 1, col='red')
```



Linear Regression Model with Less Predictors

Lets try taking out one of the highly correlated predictors and see if we can improve our model:

```
# Build Linear Regression Model
fit  <- lm(PE ~ V + RH + AP, data=df.training)
#summary(fit)

# Residuals
#plot(residuals(fit) ~ PE, data=df.training, main="Residuals Plot")
#abline(0, 0, col='red')

# Forecast the data
forecast <- forecast(fit, newdata=df.test)
accuracy(forecast, df.test)
```

```
##                        ME     RMSE      MAE         MPE     MAPE
## Training set -7.806568e-17 7.564036 5.905796 -0.02673072 1.292769
## Test set      2.167030e-02 7.489442 5.811781 -0.02413216 1.270366
##                    MASE
## Training set 0.3985527
## Test set     0.3922081
```

```
postResample(pred = forecast$mean, obs = df.test$PE)
```

```
##      RMSE  Rsquared
## 7.4894416 0.8132207
```

```
# Plot predictions versus reality in test set
#plot(df.test$PE, forecast$mean, main="Predictions of Energy Output versus Test Set", ylab="Energy Outp
#abline(0, 1, col='red')
```

Taking out the Ambient Temperature predictor, despite having high correlations to the other predictors, does not seem to help here as it reduces R^2 and increases the $RMSE$.

More Advanced Models

The above model gets a pretty good result, but I'd like to see if another model would improve the correlation coefficient or reduce the RMSE. I decided on using an SVM:

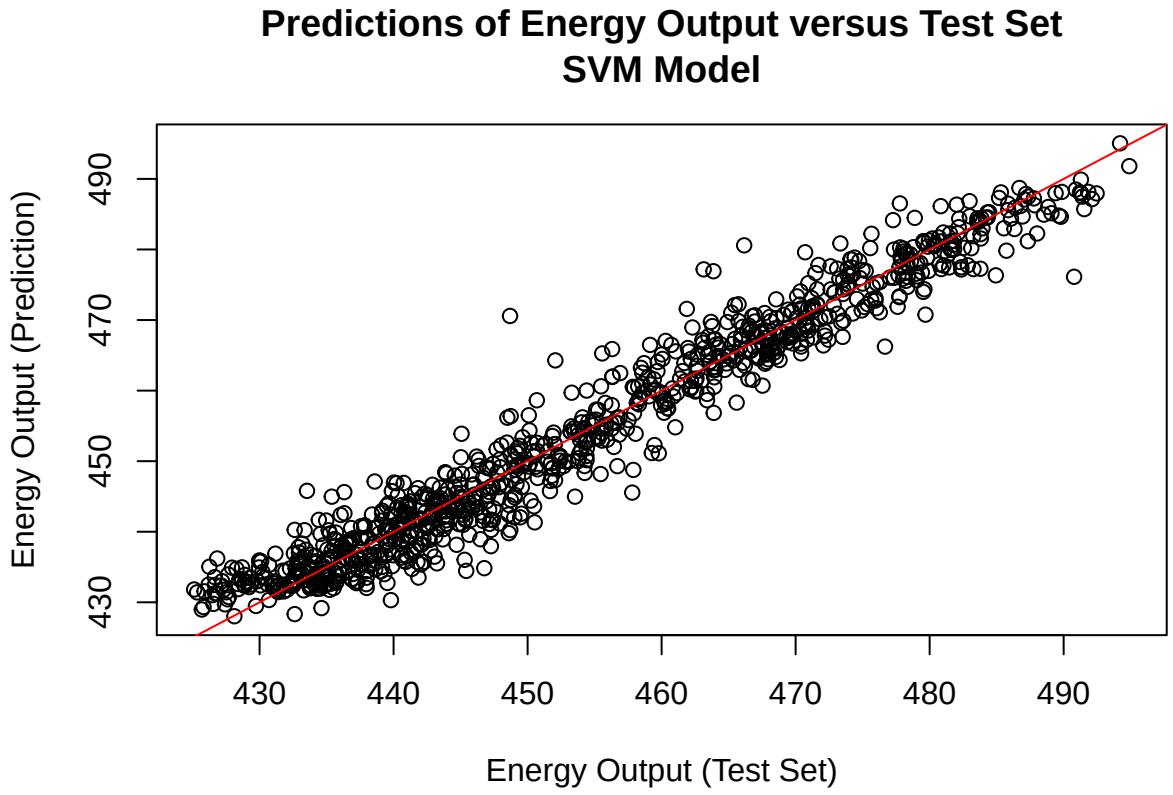
```
# Create a radial kernel SVM with some default params
library(kernlab)
svmFit <- ksvm(PE~ ., data=df, kernel="rbfdot", kpar="automatic", C=1, epsilon=0.1)
```

```
## Using automatic sigma estimation (sigest) for RBF or laplace kernel
```

```
# Predict on the test group and look at metrics
svmPredict <- predict(svmFit, df.test)
postResample(pred = svmPredict, obs = df.test$PE)
```

```
##     RMSE Rsquared
## 3.717160 0.953928
```

```
# Plot predictions versus reality in test set
plot(df.test$PE, svmPredict, main="Predictions of Energy Output versus Test Set\nSVM Model", ylab="Energ
abline(0, 1, col='red')
```

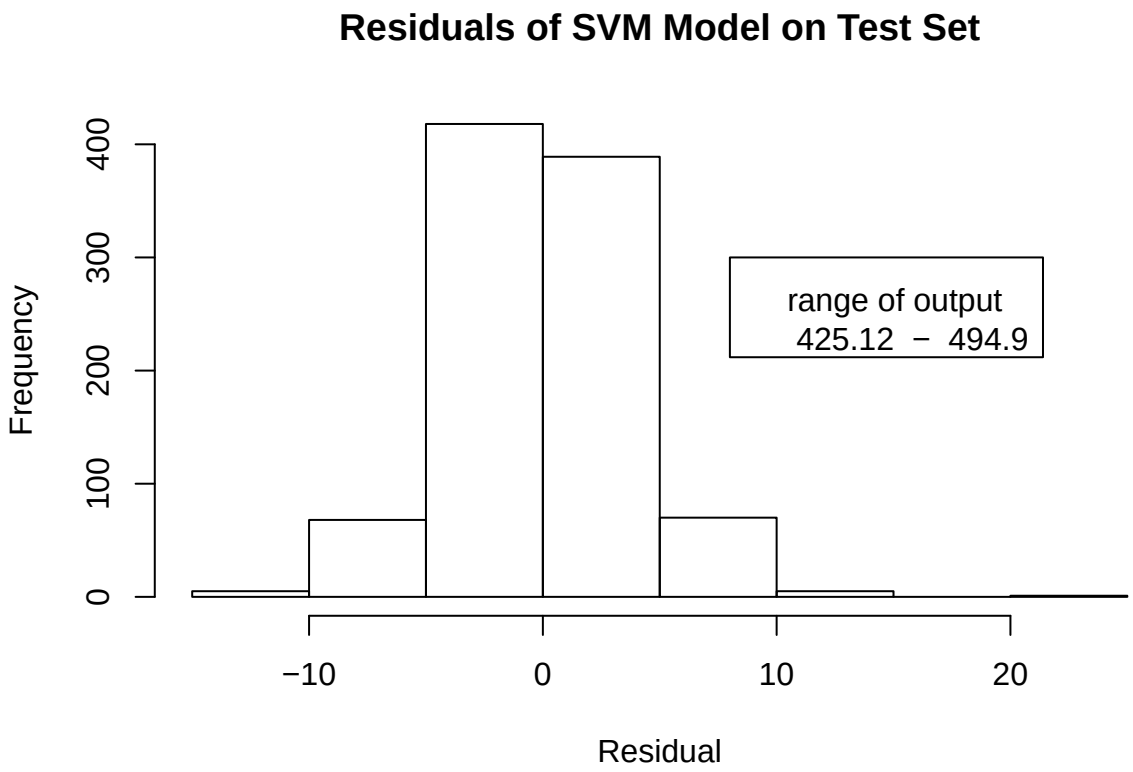


This model does a little bit better with a slightly lower RMSE and slightly higher R^2. This may indicate that the data truly is linear in nature and adding non-linearities will not help that much in improving the model.

Results

Generally, these resulting models look pretty good from an $RMSE$ and R^2 persepctive. The plots of the response from the model versus the actual response in the data do follow a straight line with the range of the residuals not being large compared to the data values:

```
hist(svmPredict - df.test$PE, main="Residuals of SVM Model on Test Set", xlab="Residual")
range <- range(df.test$PE)
legend(8, 300, paste("range of output\n", range[1], " - ", range[2]))
```



As we can see, the residuals are pretty evenly distributed around 0, with the majority of residuals between -10 and 10, which is ~7% of the output range. I would say that these models have a very good distribution of errors.

Things to Think About for Future Modeling

- What if the response (energy output) is related to these values but a 'lagged' version of the predictors? Meaning, a ΔT increase in temperature may not affect the energy output until enough time has passed for it to take effect. The lack of timing information, or data that is more 'realtime', prevents us from looking into this.
- Seasonal effects: the time of the year mat affect energy output (consumer demand, weather / climate) but once again, lack of timing data prevents us from doing this. Otherwise, time series models may have been useful.

Errors

Where can errors stem from in this model / problem:

- Faulty sensors that are recording the 4 predictor values
- As explained above, if there is any 'lag' in the relationship between the predictors and eneergy output, we cannot model it here due to a lack of data

Citations

Pınar Tüfekci, Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning methods, International Journal of Electrical Power & Energy Systems, Volume 60, September 2014, Pages 126-140, ISSN 0142-0615, http://dx.doi.org/10.1016/j.ijepes.2014.02.027. (http://www.sciencedirect.com/science/article/pii/S0142061514000908)

Heysem Kaya, Pınar Tüfekci , Sadık Fikret Gürgen: Local and Global Learning Methods for Predicting Power of a Combined Gas & Steam Turbine, Proceedings of the International Conference on Emerging Trends in Computer and Electronics Engineering ICETCEE 2012, pp. 13-18 (Mar. 2012, Dubai)