University of Minnesota, Duluth

Temperature Analysis of Szeged, Hungary

Brady Madden
Eric Erdman
Timothy Cowell
Nam Phung

STAT 5511 - Regression Analysis
Dr. Juming Pan

May 1st, 2018

Table of Contents

Abstract: This document deals with methods of predicting temperature and creating efficient models for temperature analysis. Accurate weather predictions are critical for a multitude of reasons, including knowing the appropriate time to farm, how to plan an event or day, alert people of dangerous weathers and to save individuals time by having an idea on how to dress. Whatever the reason, it is useful to study the relationship between temperature and factors that could influence it and by what degree they relate to temperature. This article covers an analysis of temperature and its correlated factors using multiple regression analysis and classification analysis.

Introduction

For the research on the analysis of temperature we wanted to understand the effects of weather conditions on temperature. Temperature plays a crucial role in medical care, food, beverages, and agriculture. Our overall health is often reliant upon temperature in many ways as well. Energy resources like natural gas and electricity also depends on weather conditions. Climate is not fixed, the fluctuation in the climate can be seen from year to year. Looking into the factors that can affect temperature is important to understanding how we can react to and control temperature.

We wanted our data to be relatable to students at the University of Minnesota Duluth.We looked for data on Minnesota's climate and did not find a good set of data that we thought we could use; however, we found a large data set from Hungary that we thought was acceptable. Hungary's climate is almost identical to Minnesota's climate based on the koppen climate classification.

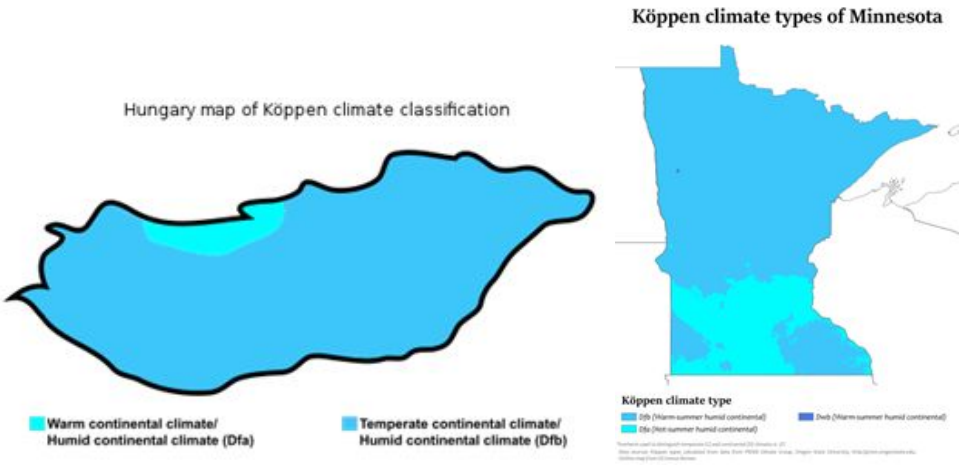


Figure 1. Koppen climate classifications of Hungary and Minnesota

Data Description

The data set for our research project is obtained from the Kaggle data science community. The dataset contained 95,936 data entries collected from 2006 to 2016. The name, description, and type of the attributes in the dataset is given in the table below:

Name	Type	Labels	Description
Temp	Numeric, Continuous		Hourly temperature in a day,

			measured in Celsius.
Apparent.Temp	Numeric, Continuous		Temperature that human perceive. Measured in Celsius.
Formatted.Date	String		Date and time the data entry was collected
Summary	String, Categorical	Contains 27 labels	Short summary of the current weather
Precip.Type	String, Categorical	rain, snow	Type of precipitation
Humidity	Numeric, Continuous		Level of humidity
Wind.Speed	Numeric, Continuous		Speed of the wind in km/h
Wind.Bearing.deg	Numeric, Continuous		Direction of the wind
Visibility	Numeric, Continuous		Visibility measured in km
Loud.Cover	Numeric, Categorical	0	Rating of the cover of cloud. This data seemed to be corrupted since there was only 1 label
Pressure	Numeric, Continuous		Pressure measured in millibars
Daily.Summary	String, Categorical	Contains 214 labels	Full summary at the current hour

Table 1. Variable names and description

The dataset contained many interesting features that we could use for our prediction. However, in order to use these attributes, we had to preprocess the dataset to extract this information. Firstly, it seemed that the attribute Loud.Cover was corrupted since there was only 1 unique value in that column, we decided to drop that variable. The Formatted.Date attribute contained many interesting data that could be useful for our analysis, such as day, month, hour, and year. Therefore, we processed the text strings in Formatted.Data column to extract these attributes and conducted a graphical analysis to see the whether a correlation between temperature and these attributes exists or not. The graph below shows the temperature in the year 2014. It seemed like there is a correlation between these attributes and the temperature.

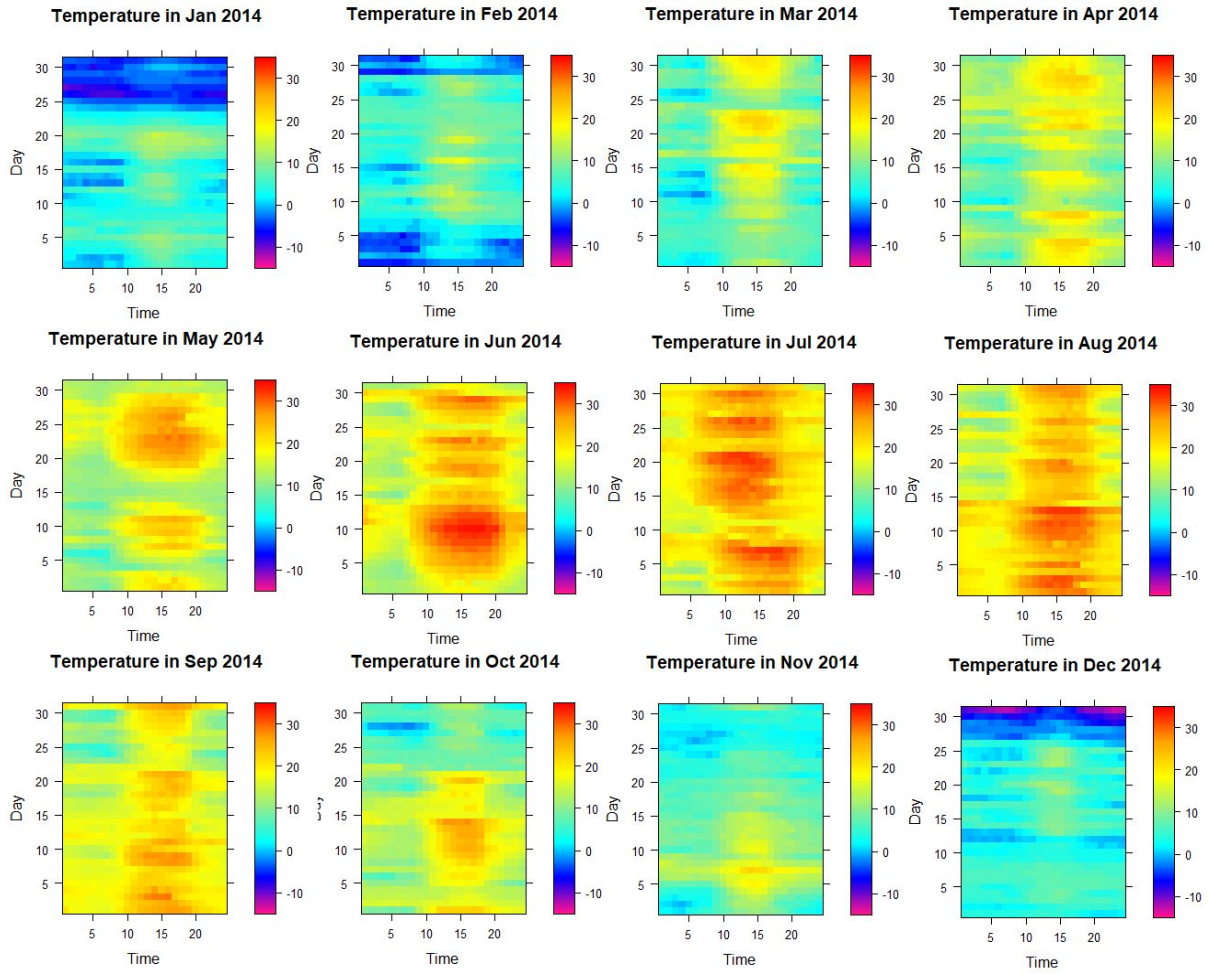


Figure 2. Temperature plot over 12 months in 2014

However, this plot was only for the year of 2014, we wanted to see whether there exists a systematic pattern in the relationship between temperature, month, and day. Therefore, we created a 3D plot with 1000 data samples. The plot showed that there seems to exist a certain pattern which describe the relationship between these 3 variables. Therefore, it is reasonable to include these variables in our analysis.

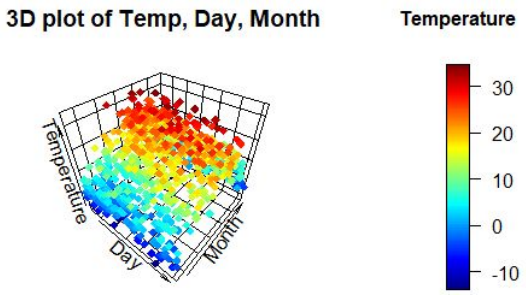


Figure 3. 3D plot of the temperature against the day and month

Other attributes that we decided to process were Precip.Type and Summary. For Precip.Type, the procedure was simple, we map the list of unique values from (rain, snow) to (1, 2) in order to include in our predictive models. As for the variable Summary, the attribute contains 27 different labels, therefore, we mapped each label into an appropriate group to reduce the number of the labels down and assign a new name for this attribute: Processed.Summary. Each group will contain labels that are somewhat similar to each other. The mapping of these labels is shown in the following table.

Group	Labels
1	Clear
2	Partly Cloudy, Humid and Partly Cloudy, Windy and Partly Cloudy, Dry and Partly Cloudy, Dangerously Windy and Partly Cloudy
3	Mostly Cloudy, Breezy and Mostly Cloudy, Breezy and Partly Cloudy, Humid and Mostly Cloudy, Windy and Mostly Cloudy, Dry and Mostly Cloudy
4	Overcast, Breezy and Overcast, Windy and Overcast, Humid and Overcast
5	Foggy, Windy and Foggy, Breezy and Foggy
6	Breezy, Dry, Windy, Light Rain, Drizzle, Windy and Dry, Breezy and Dry, Rain

Table 2. Grouping of labels in attribute Summary to create new attribute Processed.Summary

We created a scatter plot of the Temp attribute against the new attribute Processed.Summary to investigate the relationship between these 2 variables. The graph showed that there might be some relationship between these attributes. However, the correlation between them might be low. Nonetheless, we still decided to keep this attribute instead of dropping it because it might be useful later on in our analysis.

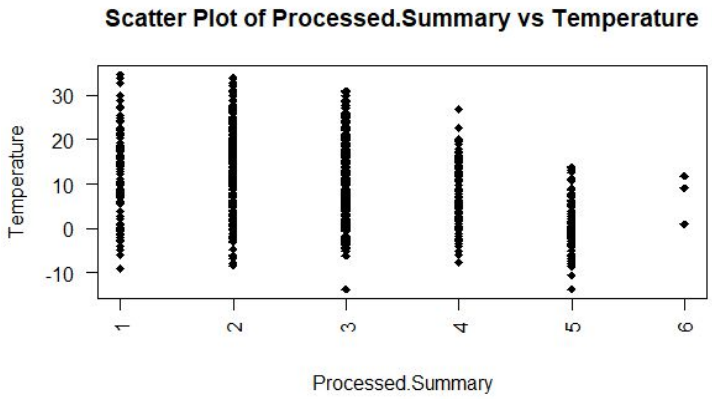


Figure 4. Scatter plot of the Temperature against Processed.Summary

Because our project includes classification analysis, we processed the Temp attribute into a categorical variable that can be used for classification models by discretizing the variable. We created 8 class labels as follows.

```
> unique(data.discretized$Temp)
[1] [9.04,16.8)   [1.33,9.04)
[3] [16.8,24.5)   [24.5,32.2)
[5] [-6.39,1.33)  [32.2,39.9]
[7] [-14.1,-6.39) [-21.8,-14.1)
8 Levels: [-21.8,-14.1) ... [32.2,39.9]
```

Figure 5. Class labels of the discretized Temp attribute

Each temperature will be mapped to an appropriate temperature interval. We will discuss the reason for switching from regression analysis to classification analysis in the section on classification models. The distribution of the class labels is shown the figure below.

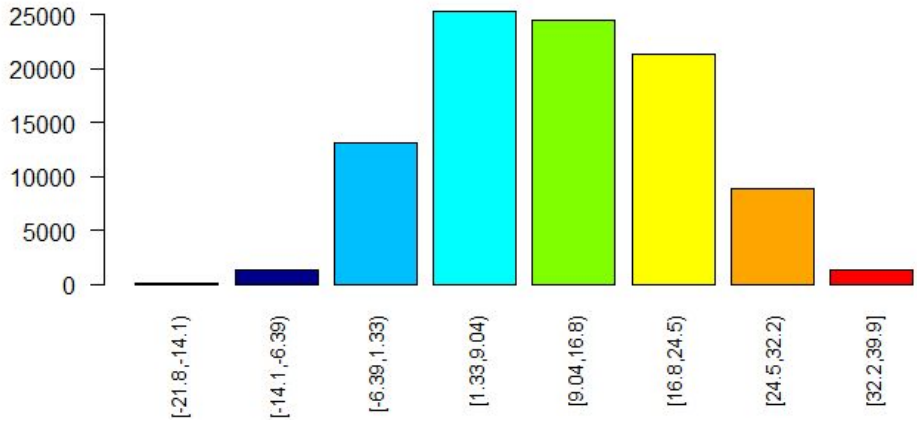


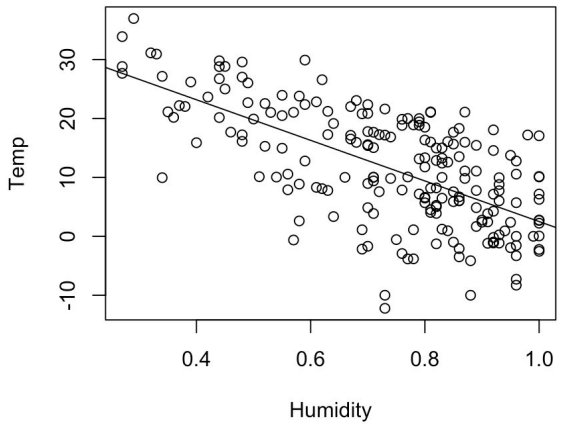
Figure 6. Distribution of the discretized Temp attribute

Finally, we also computed the correlation between the regressors and the response variable Temp to pick variables that would be included in our predictive models. Some attributes that have higher correlation with the Temp are: Humidity, Precip.Type, Visibility, etc. It is also worth notice that although Processed.Summary has a high correlation than other attributes such as Month and Day. This does not guarantee that Month and Day are less important than Processed.Summary. In fact, these attributes are much more important than Processed.Summary. This will be shown in the section on classification analysis.

	Apparent.Temp	Humidity	Wind.Speed	Wind.Bearing.deg	Visibility	Pressure	Precip.Type	Day	Month	Year	Hour	Summary	Processed.Summary	Temp
Apparent.Temp	1.00	-0.60	-0.06	0.03	0.38	0.00	-0.57	0.00	0.16	0.02	0.16	-0.19	-0.35	0.99
Humidity	-0.60	1.00	-0.22	0.00	-0.37	0.01	0.23	0.01	0.06	0.04	-0.29	0.16	0.37	-0.63
Wind.Speed	-0.06	-0.22	1.00	0.10	0.10	-0.05	-0.07	-0.01	-0.10	-0.01	0.07	0.09	0.09	0.01
Wind.Bearing.deg	0.03	0.00	0.10	1.00	0.05	-0.01	-0.04	0.00	-0.02	-0.03	0.00	-0.01	-0.02	0.03
Visibility	0.38	-0.37	0.10	0.05	1.00	0.06	-0.32	-0.02	-0.07	0.11	0.00	-0.18	-0.49	0.39
Pressure	0.00	0.01	-0.05	-0.01	0.06	1.00	0.01	-0.02	-0.02	0.02	0.00	-0.13	0.07	-0.01
Precip.Type	-0.57	0.23	-0.07	-0.04	-0.32	0.01	1.00	0.01	-0.15	-0.04	-0.06	0.12	0.20	-0.56
Day	0.00	0.01	-0.01	0.00	-0.02	-0.02	0.01	1.00	0.01	0.00	0.00	0.02	0.00	0.00
Month	0.16	0.06	-0.10	-0.02	-0.07	-0.02	-0.15	0.01	1.00	-0.01	0.00	0.00	-0.02	0.15
Year	0.02	0.04	-0.01	-0.03	0.11	0.02	-0.04	0.00	-0.01	1.00	0.00	-0.03	0.06	0.02
Hour	0.16	-0.29	0.07	0.00	0.00	0.00	-0.06	0.00	0.00	0.00	1.00	-0.09	-0.01	0.17
Summary	-0.19	0.16	0.09	-0.01	-0.18	-0.13	0.12	0.02	0.00	-0.03	-0.09	1.00	0.08	-0.19
Processed.Summary	-0.35	0.37	0.09	-0.02	-0.49	0.07	0.20	0.00	-0.02	0.06	-0.01	0.08	1.00	-0.35
Temp	0.99	-0.63	0.01	0.03	0.39	-0.01	-0.56	0.00	0.15	0.02	0.17	-0.19	-0.35	1.00

Figure 7. Correlation between attributes in the dataset

Simple Linear Regression




```
Response: Temp
            Df  Sum Sq Mean Sq F value    Pr(>F)
Humidity     1  8219.7  8219.7   147.1 < 2.2e-16 ***
Residuals  198 11063.8    55.9
---
```

Figure 8. Temperature~Humidity plot and Anova response Temperature and regressor humidity

We created a scatter plot of the temperature relative to the humidity using 200 random data points within our sample. Humidity and Temperature seemed to correlate the best, which points to why we used these two variables. We fit a regression line to the data, and ran an ANOVA test which showed that Humidity is very important to temperature, as the Pr(>F) had a very low value.

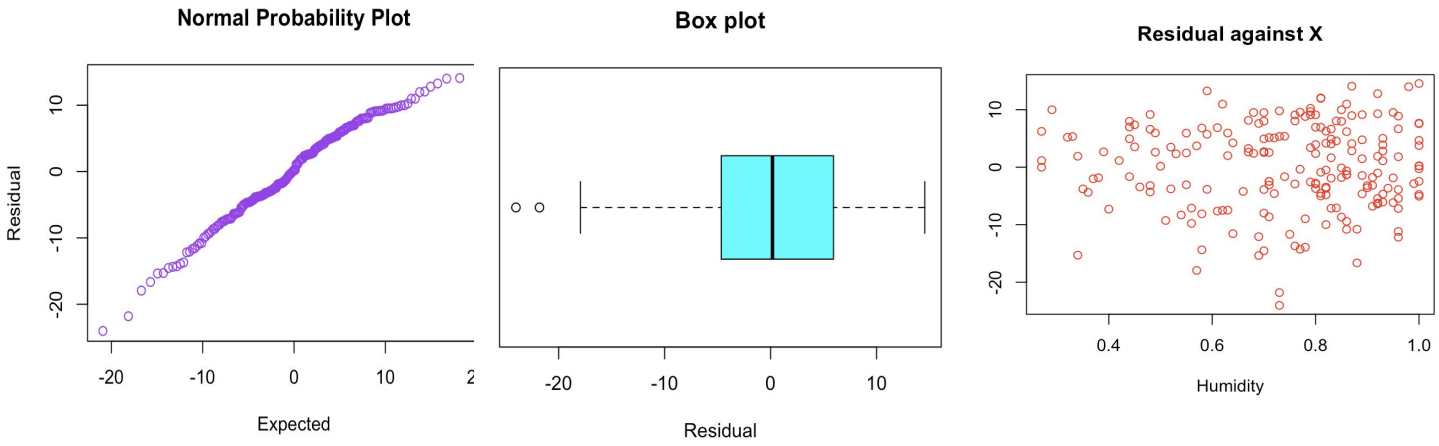


Figure 9. Normal probability plot, box and whisker plot, and residual plot

We called for a normal probability plot, box and whisker plot, and residual plot, to make sure that our data followed all the assumptions needed to accurately run a simple linear regression. Our normal probability plot and residual plot showed no issues with steady variance, and assumption of normality, and the box and whisker plot only showed two outliers.

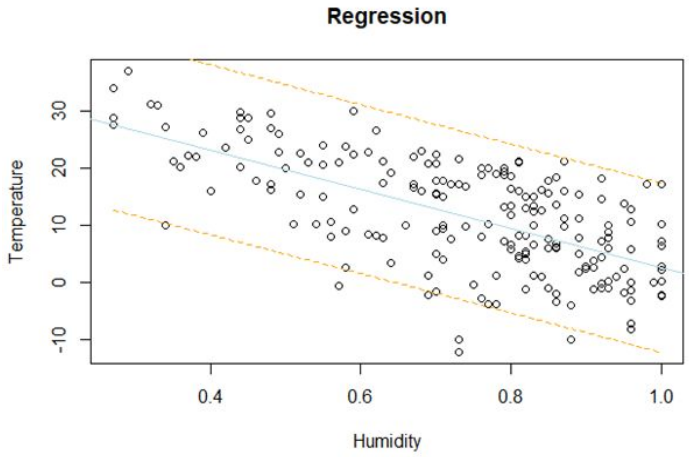


Figure 10. Prediction interval Temperature~Humidity

The Prediction interval was run using new response variables that ranged from the minimum value of humidity and the maximum value of humidity at intervals to form a list of 200 variables. The plot shows that when temperature is plotted against humidity the values from the data set fall within the upper and lower intervals of the prediction interval. With the exception of a couple outliers. From this we could predict a range for the temperature within 95%. There are other factors that should be accounted for when determining the temperature; however, just using humidity alone gets an accurate range for the temperature.

Transformations

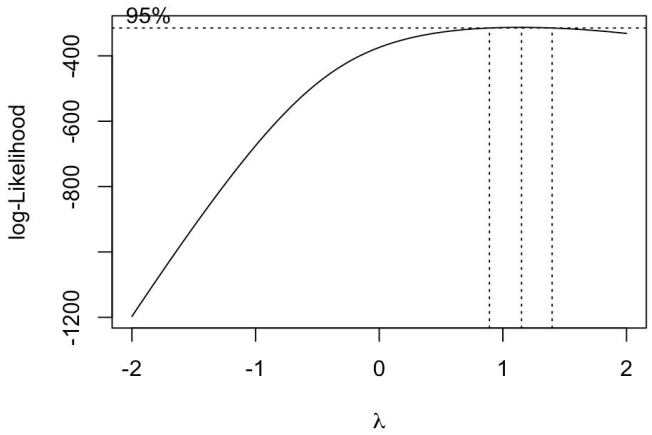


Figure 11. Box-cox plot

We wanted to check to see if a transformation may help our simple linear regression model at all, so we ran a box-cox. The box-cox showed that a transformation wasn't necessary as the confidence interval covers 1, but we performed one anyway, just to see how much it would change our outcomes. As you can see below, there wasn't too much of a change.

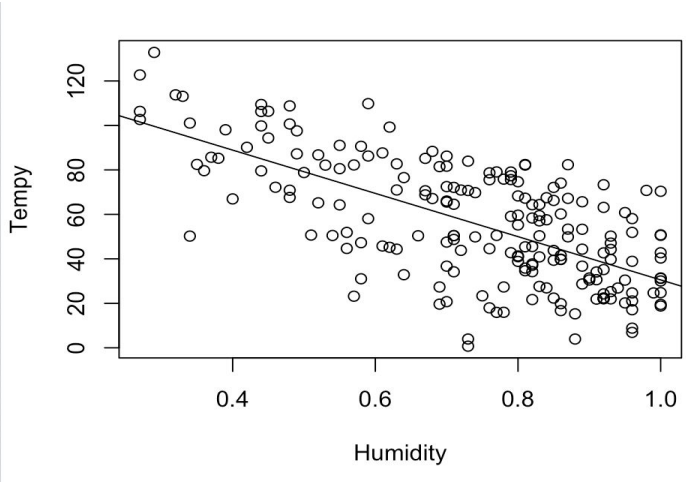


Figure 12. Temperature~Humidity plot

Multiple Linear Regression

Full Model Analysis

The start of our analysis on multiple linear regression starts with the full model with parameters, humidity, wind speed, wind bearing speed, visibility, and pressure. An analysis of variance was performed on the model as a whole. The results show that the predictor variables humidity,

visibility and wind speed are all significant to the model. There is an F-statistic of 32.97 on 5 and 194 degrees of freedom. The p-value is much below zero, which implies that at least that one of the regressors contributes significantly to the model. With this information, we know at least one regressor variables are significant and we will look into which variables contribute the most to the model.

```
Analysis of Variance Table

Response: Temp
                Df  Sum Sq Mean Sq  F value  Pr(>F)
Humidity         1  8219.7  8219.7 152.9575 < 2e-16 ***
Wind.Speed       1   309.6   309.6   5.7617 0.01732 *
Wind.Bearing.deg 1     0.1     0.1   0.0014 0.96967
Visibility       1   313.6   313.6   5.8348 0.01664 *
Pressure         1    15.3    15.3   0.2844 0.59445
Residuals      194 10425.3    53.7
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 7.331 on 194 degrees of freedom
Multiple R-squared:  0.4594,    Adjusted R-squared:  0.4454
F-statistic: 32.97 on 5 and 194 DF,  p-value: < 2.2e-16
```

Reduced Model Analysis

We used the analysis of variance table of the full model to help us break the model into a couple of different reduced models, which we used to test the significance and contribution of the regressor variables. The first reduced model was made with humidity, wind speed and visibility. Running this reduced model against our full model shows an F-statistic of 0.1657 and a large p-value of 0.8474. These results show that wind bearing and pressure are not important regressors given that humidity, visibility and wind speed are in the model.

```
Analysis of Variance Table

Model 1: Temp ~ Humidity + Wind.Speed + Visibility
Model 2: Temp ~ Humidity + Wind.Speed + Wind.Bearing.deg + Visibility +
    Pressure
  Res.Df   RSS Df Sum of Sq      F Pr(>F)
1    196 10443
2    194 10425  2    17.809 0.1657 0.8474
```

The second reduced model was made with the parameters, wind bearing degree and pressure. Running this reduced model against the full model shows an F-statistic of 54.834 and a p-value much lower than 0.001. These statistics show that humidity, wind speed and visibility are important variables given that wind bearing and pressure are in the model. This shows that humidity, wind speed and visibility contribute significantly to the model.

```
Analysis of Variance Table

Model 1: Temp ~ Wind.Bearing.deg + Pressure
Model 2: Temp ~ Humidity + Wind.Speed + Wind.Bearing.deg + Visibility +
    Pressure
  Res.Df   RSS Df Sum of Sq      F    Pr(>F)
1    197 19265
2    194 10425  3    8840.1 54.834 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The next step was running an analysis of variance on our reduced model of humidity, wind speed and visibility to get a better understanding of how much better these variables are at predicting temperature. The analysis of variance shows us that all of the variables are now significant in the model. It also produced an F-statistic of 55.31 on 3 and 196 degrees of freedom and a very small p-value below 0.001. The R-squared adjusted is also seen to have increased by 0.0048 when compared to the full model. This slight increase in R-squared adjusted shows us that taking out the variables wind bearing and pressure can get similar results in predicting temperature with less variables. This is important, because having less variables leads to easier computations. This can be valuable, especially if hand calculations are necessary. This model is slightly better in residual standard error as well, which is another indication that this model is a better representation than the full model.

```
Call:
lm(formula = Temp ~ Humidity + Wind.Speed + Visibility)

Residuals:
     Min       1Q   Median       3Q      Max
-21.6592  -4.1580   0.8081   5.5149  15.6857

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  34.79231    3.32300  10.470   <2e-16 ***
Humidity    -33.05740    2.99825 -11.026   <2e-16 ***
Wind.Speed   -0.21275    0.08977  -2.370   0.0188 *
Visibility    0.31082    0.12863   2.416   0.0166 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 7.299 on 196 degrees of freedom
Multiple R-squared:  0.4584,    Adjusted R-squared:  0.4502
F-statistic: 55.31 on 3 and 196 DF,  p-value: < 2.2e-16
```

Best Subsets of Variables

Variable selection is important in producing the best possible model. An exhaustive search was conducted for the best subset of variables for the k amount of regressors. The exhaustive search shows us which variables would be best to choose if we wanted a certain amount regressors. The R-squared adjusted for having just humidity in the model is 0.4234. The R-squared adjusted for

having humidity and visibility in the model is 0.4373. The R-squared adjusted for having humidity, visibility and wind speed in the model is 0.4502. The R-squared adjusted for having humidity, visibility, wind speed and wind bearing degree in the model is 0.4475. The R-squared adjusted for the full model is 0.4454. Analyzing the exhaustive search showed us that the subset with three variables is the best model to use for predicting temperature. It is interesting to note that this exhaustive search is consistent with the reduced model anova that was performed and it is important to have more reassurance in the selection of the best regressor variables.

	Humidity	Wind.Speed	Wind.Bearing.deg	Visibility	Pressure
1 (1)	*				
2 (1)	*			*	
3 (1)	*	*		*	
4 (1)	*	*	*	*	
5 (1)	*	*	*	*	*

Table 3. Exhaustive Search for the Best Variable

Temperature Prediction as A Classification Problem

In real world application, it is often hard to predict the correct temperature given attributes such as humidity, wind speed, wind degree, and so on. Because of this, we presented an alternative approach to the regression analysis discussed above. The data set is processed by discretizing the Temp attribute as mentioned before.

We used the new dataset to fit 2 different classification models and compare them by the end to see which model performs better without overfitting or underfitting the training data. The first model is a K-Nearest Neighbor Model (KNN Model). The KNN model classifies a data point based on the dominating class among the k nearest neighbor to the data point. One problem aroused when fitting our KNN model is to find an optimal k value. If the k value is too low, our model will be susceptible to noise. If the k value is too high, the model might take into consideration data points that are too far away from it. The figure below shows the attributes that we used to predict the temperature class.

$$Temp \sim Humidity + WindDirection + Pressure + Precip.Type + Month + Day + Hour$$

Figure 13. Attributes used in KNN model

In order to find the optimal k value, we used the data to fit the KNN model with different K values to find the optimal value. In order to get the best approximation of the performance of the model with different K values, we also applied K-Fold cross validation method for each K value. A value K is considered to be optimal if it minimizes the misclassification rate when the model is fitted using the training data. The figure below shows that the optimal K value is in the range of [8,12].

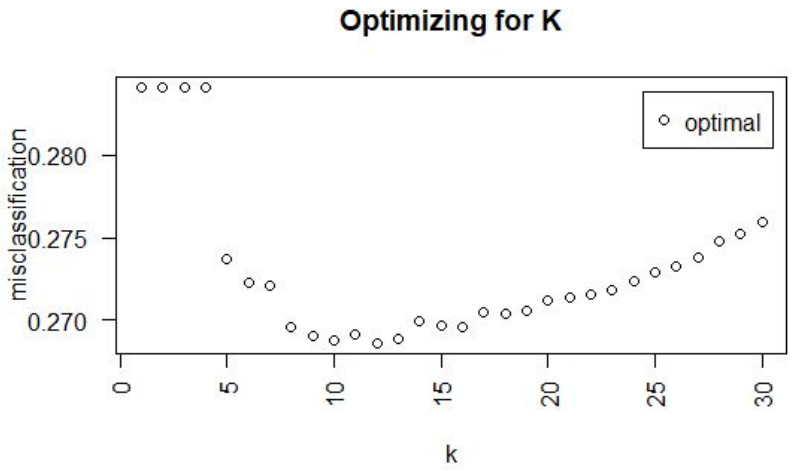


Figure 14. Misclassification rate of the KNN model when fitted against different K value

The second classification model that we proposed is called the Random Forest Model. Random forest is an ensemble model in which it combines multiple decision tree models into a compound model that makes prediction based on the prediction of the majority of the smaller models. A decision tree model makes prediction based on the split at each node in the tree. In the figure below is an example of a decision tree model that makes prediction of whether a person should play tennis or not given the current weather report.

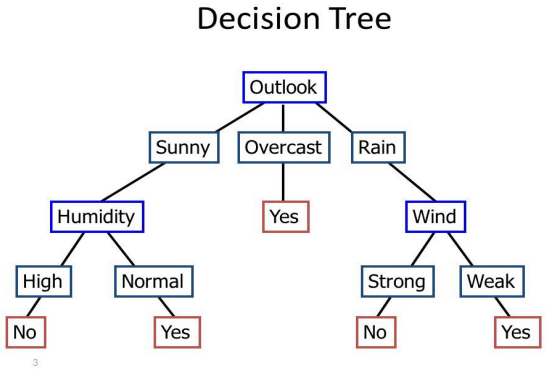


Figure 15. Example of a decision tree model

In order to measure the performance of the decision tree, a coefficient named GINI index is computed to measure the level of impurity of each leaves node (the red nodes shown in the example above). A decision tree model is considered to be ideal when it has a GINI index of 0. The formula for computing the GINI index is as follows:

$$GINI(t) = \sum_j [p(j|t)]^2$$

$$p(j|t) \text{ is the frequency of } j \text{ at node } t$$

For our random forest model, we built 500 decision trees to make a prediction on the weather. The figure below shows the regressors that we used to make a temperature prediction.

$$Temp \sim Humidity + WindDirection + Pressure + Precip.Type + Month + Day + Hour + Processed.Summary$$

Figure 16. Regressors used to fit the Random Forest model

In order to determine which variables will be useful in our random forest model, we performed a test on the decrease of GINI index to check the importance of each variable. The graph below shows the contribution of the variables to the model in decreasing order, starting from Month, Humidity, Pressure, Precip.Type, Wind.Bearing.deg, Day, Hour, Processed.Summary. Although the last variable was not very important, we decided to include it in our final model.

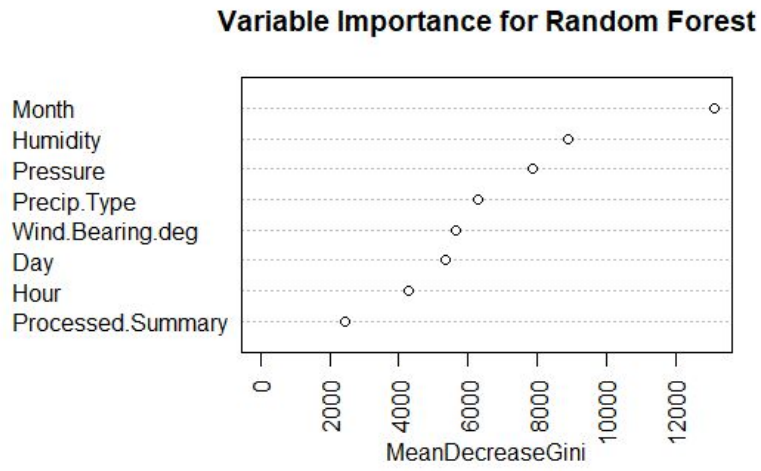


Figure 17. The importance of each variable in the random forest model

We conducted a test on the performance of the 2 classification models to pick the superior model. In order to do that, we partitioned the original dataset into 2 portions: training data set

and test data set. For example, the dataset is partitioned into 80% used for fitting the model and 20% for testing the performance of the model. The result of the model's performance given different amount of training data is shown in the table and the graph below:

Dataset		**KNN Model**	**Random Forest Model**
Training portion	**Testing portion**	**Accuracy**	**Accuracy**
5%	95%	0.6354839	0.7132982
10%	90%	0.6611306	0.7301692
20%	80%	0.6813900	0.7574822
30%	70%	0.6937429	0.7710703
40%	60%	0.7007748	0.7799764
50%	50%	0.7131629	0.7918195
60%	40%	0.7170554	0.7955440
70%	30%	0.7256523	0.8043153
80%	20%	0.7263915	0.8073796

Table 4. Performance of 2 models when training & testing with different amount of data

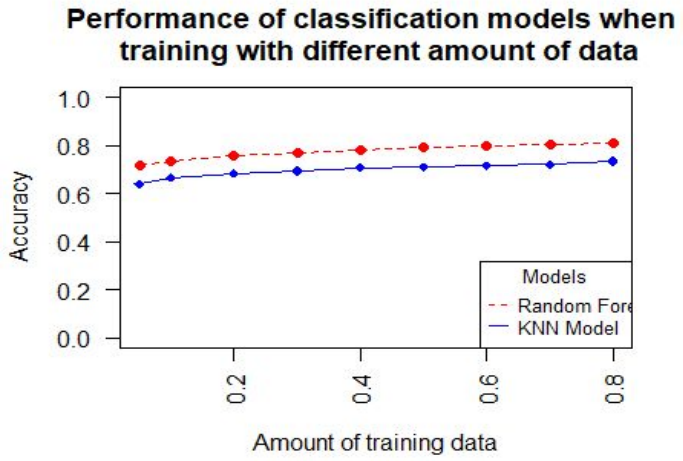


Figure 18. Performance of 2 models when training & testing with different amount of data

Besides looking at the overall accuracy, we also conducted a graphical analysis of the performance of 2 models to see the accuracy of the models on different class labels. We computed 2 confusion matrix when testing the models using 80% of the original dataset and created a level plot to see the accuracy for each class label.

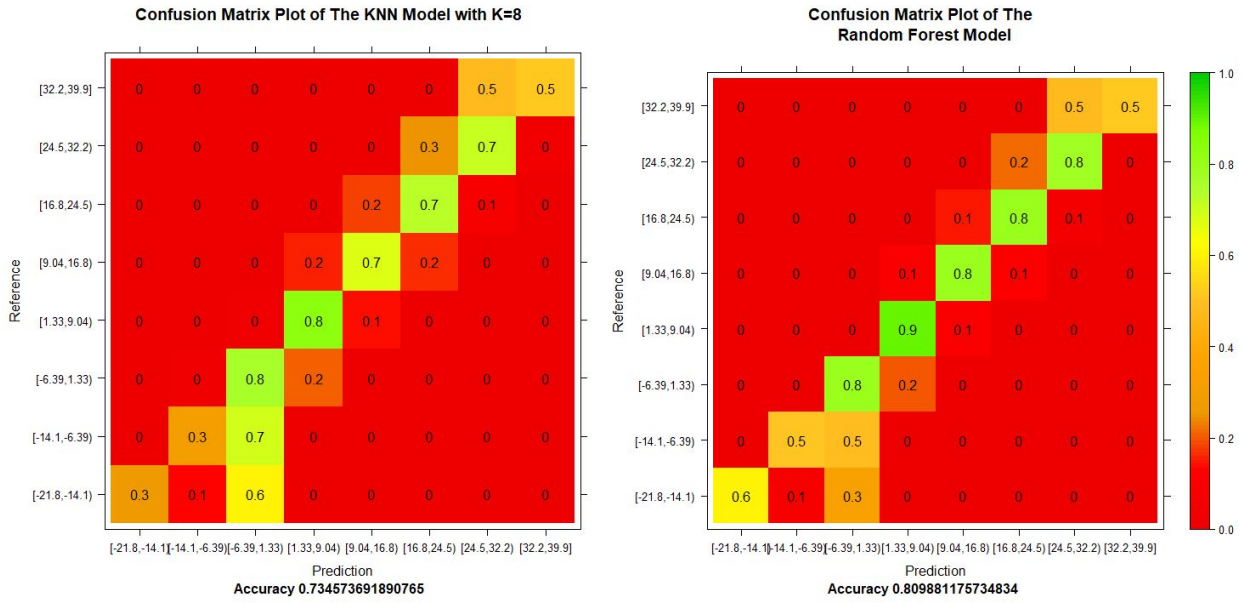


Figure 19. Confusion matrix plot of 2 models when training with 80% data

Based on the graph above, it seems like both models performed with relatively high accuracy when the data entry has a temperature in the range $[-6.39,\ 32.2)$. We suspected the reason for the lower accuracy of the marginal class labels was because of lacking data. There were over 95,936 data entries for temperature in the interval $[-6.39,\ 32.2)$, which occupied 96.99% of the original dataset. The confusion matrix plot also showed that the misclassification class labels were not far away from the actual class labels. Overall, it seemed that none of these classification models were running into underfitting or overfitting problem. We believed that it is reasonable to conclude that the Random Forest is the better model due to its high accuracy in all of the tests.

Discussion & Future Works

Our research is only looking at one climate condition. The koppen climate classification lists twenty five different climates. Further research into different climates and the differences between the climates, could give us a better understanding of the different effects on temperature. Another future possibility is a controlled experiment where you can isolate certain factors and control the variables. Using various different models to accurately look at the different conditions as well can improve predictions and our understanding. These types of studies on temperature could help with saving energy on cooling down homes or heating them up

in winter. Also studies over time could help with understanding a way to help slow down global warming and climate change.

As for the classification analysis, we can look into some other classification models such as logistic regression (Logit), artificial neural networks (ANN), and support vector machine (SVM) to analyze their performance against the 2 models presented in this paper. Instead of finding an optimal model, we could try to build multiple models with good performance and use the ensemble modeling technique to build a compound model like the random forest model. Moreover, if we construct a model with better performance, we can increase the number of class labels to be higher than 8. Increasing the number of class labels would make the prediction more accurate by decreasing the temperature interval of each class.

Bibliography

NorbertBudincsevity. Weather in Szeged 2006-2016 | Kaggle, 8 Jan. 2017, www.kaggle.com/budincsevity/szeged-weather.

Appendices

Appendix A: R code for the linear regression analysis
Appendix B: R code for the figures and classification analysis
Appendix C: what items the people in group worked on

Appendix A

```
# Code for Simple Linear Regression section:
EricsD2 <- read.csv("~/Desktop/EricsD2.txt", sep="")
View(EricsD2)
attach(EricsD2)
plot(x,y,pch=1)
lm(y ~ (x^(.5)))
lm(y~x)
plot((x^.5),y,pch=19)
fitreg<-lm((x^(.5)) ~ y)
abline(fitreg)
res<-model$residuals
plot(x,res,pch = 19,main="Regression Residuals")
qqplot(qnorm(ppoints(length(res)),mean(res),sd(res)),res,
+         xlab="Expected",ylab="Residual",
+         main="Normal Probability Plot",col="Purple")
weatherHistory <- read.csv("~/Desktop/weatherHistory.csv")
View(weatherHistory)
weatherHistory.2 <- read.csv("~/Desktop/weatherHistory 2.csv")
View(weatherHistory.2)
attach(weatherHistory.2)
fit=lm(Humidity~Temperature..C.)
abline(fit)
plot(Temperature..C.,Humidity)
RegressProj <- read.csv("~/Desktop/RegressProj.csv")
View(RegressProj)
attach(RegressProj)
Rproj=lm(Temp~Humidity, data=RegressProj)
library(MASS)
boxcox(Rproj)
RegressProj$Temp = (RegressProj$Temp - min(RegressProj$Temp)) /
(max(RegressProj$Temp) - min(RegressProj$Temp))
boxcox(RegressProj$Temp)

#simple linear regression code-tim
#prediction interval
data = read.csv(file.choose(), sep=',')
attach(data)

fitreg<-lm(Temp~Humidity,data=data)
anova(fitreg)
x<-(Humidity)
y<-(Temp)
x2<-(x^2)
```

```
xbar<-(sum(x)/200)
nxbar<-(200*(.73985^2))
Sxx<-(sum(x2-nxbar))
xstar<-seq(min(data$Humidity),max(data$Humidity),length.out = 200)
new= data.frame(Humidity=xstar)
pi<-predict(fitreg,newdata = new, interval="predict", level = 0.95)
lower<-pi[,2]
upper<-pi[,3]

plot(Temp~Humidity, xlab="Humidity", ylab="Temperature",
main="Regression")
abline(fitreg, col="lightblue")

lines( xstar,lower, col="orange", lty=2)
lines(xstar,upper, col="orange", lty=2)
```

Code for Multiple Linear Regression section:

```
attach(weatherHistorySampled_csv)
fitReg=lm(Temp~Humidity+Wind.Speed+Wind.Bearing.deg+Visibility+Pressure)
anova(fitReg)
summary(fitReg)
reduced1=lm(Temp~Humidity+Wind.Speed+Visibility)
reduced2=lm(Temp~Wind.Bearing.deg+Pressure)
summary(reduced1)
summary(reduced2)
anova(reduced1)
anova(reduced2)
anova(reduced1,fitReg)
anova(reduced2,fitReg)
reduced3=lm(Temp~Humidity+Wind.Speed+Visibility+Wind.Bearing.deg)
reduced4=lm(Temp~Humidity+Visibility)
summary(reduced3)
summary(reduced4)
```

Appendix B

```
# Required packages
usePackage <- function(p) {
  if (!is.element(p, installed.packages()[,1]))
    install.packages(p, dep = TRUE)
  require(p, character.only = TRUE)
}
usePackage("plyr")
usePackage("PerformanceAnalytics")
usePackage("rpart")
usePackage("rpart.plot")
usePackage("arules")
usePackage("caret")
usePackage("e1071")
usePackage("DMwR")
usePackage("plot3D")
usePackage("randomForest")
usePackage("kknn")
usePackage("leaps")

# Variables contain the dataframe
# data - processed dataframe used for regression problem
# data.discretized - discretized dataframe based on k-cluster used for classification problem (5 groups)
# data.discretized2 - discretized dataframe based on interval used for classification problem (15 groups)

# Read the weather data
data = read.csv('WeatherHistory.csv', sep=',')

#########################################################################
# DATA PREPROCESS
#########################################################################

# Extract day
data$Day = as.numeric(sapply(data$Formatted.Date, substring, 9, 10))

# Extract month
```

```
data$Month = as.numeric(sapply(data$Formatted.Date, substring, 6, 7))

# Extract year (used for plotting heatmap)
data$Year = as.numeric(sapply(data$Formatted.Date, substring, 1, 4))

# Extract hour
data$Hour = as.numeric(sapply(data$Formatted.Date, substring, 12, 13))

# Drop unused columns
data = data[, !(colnames(data) %in% c("Formatted.Date", "Loud.Cover", "Daily.Summary"))]

# Take out rows with null class label in column Precip.Type
# Required packages:
#   + plyr
#   + PerformanceAnalytics
data = data[data$Precip.Type != 'null', ]

# Map categorical variable in Precip.Type to numerical value
# 1 = rain
# 2 = snow
data$Precip.Type <- mapvalues(data$Precip.Type,
                              from=c("rain","snow"),
                              to=c(1, 2))

# Map categorical variable in Summary to numerical value
data$Summary <- mapvalues(data$Summary,
                              from=as.vector(unique(data$Summary)),
                              to=seq(1, length(as.vector(unique(data$Summary))), 1))
# 1=clear, 2=partly cloudy, 3=mostly cloudy, 4=overcast, 5=foggy, 6=others
data$Processed.Summary <- mapvalues(data$Summary,
                            from=seq(1, length(as.vector(unique(data$Summary))), 1),
                            to=c(2, 3, 4, 5, 3, 1, 3, 4, 3, 2, 5, 4, 5, 2, 6, 2, 3, 2, 6, 6, 4, 6, 6, 6, 3, 6, 6))

# Convert variables to numeric variables
columns = names(data)
for (column in columns) {
  data[[column]] = as.numeric(as.character(data[[column]]))
}
data = data
```

```
# Standardize all columns except response column & columns with categorical
# values
excluding_columns = c("Precip.Type", "Day", "Month", "Year", "Hour", "Summary",
"Processed.Summary", "Temp")
tmp = as.data.frame(scale(data[, !names(data) %in% excluding_columns]))
tmp[excluding_columns] = data[excluding_columns]
data = tmp
rm(tmp)
rm(excluding_columns)

# Discretize the temperature column & turn into a classification problem
# Discretize temperature column by breaking into 8 groups with same interval
data.discretized = data
data.discretized$Temp = discretize(data.discretized$Temp,
                                   method = "interval",
                                   breaks = 8)
data.discretized.targets = unique(as.vector(data.discretized$Temp))

#####################################################################
# PLOTTING
#####################################################################

data.plot = data[sample(nrow(data), 1000), ]

# Do a correlation plot & examine correlations among variables (only
# use first 500 data entries)
chart.Correlation(data.plot,
                  method="spearman",
                  histogram=TRUE,
                  pch=16)

# Scatter plot of humidity vs temperature
plot(data.plot$Temp~data.plot$Humidity,
    main="Scatter Plot of Humidity vs Temperature",
    xlab="Humidity",
    ylab="Temperature")

# Scatter plot of month vs temperature
```

```
plot(data.plot$Temp~data.plot$Month,
    main="Scatter Plot of Month vs Temperature",
    xlab="Month",
    ylab="Temperature")

# Scatter plot of summary vs temperature
plot(data.plot$Temp~data.plot$Summary,
    main="Scatter Plot of Summary vs Temperature",
    xlab="Summary",
    ylab="Temperature")

# Scatter plot of summary vs temperature
plot(data.plot$Temp~data.plot$Processed.Summary,
    main="Scatter Plot of Processed.Summary vs Temperature",
    xlab="Processed.Summary",
    ylab="Temperature",
    pch = 18)

year = 2014
plots = list()
for (month in 1:12) {
  tmp = data[data$Year == year & data$Month == month, ][, c("Day", "Temp")]
  tmp = tmp[order(tmp$Day),]
  tmp = matrix(as.matrix(tmp$Temp), nrow = 31, ncol=24, byrow = TRUE)

  breaks = seq(-20, 40, by=0.01)
  x = levelplot(t(tmp), xlab="Time",
                ylab="Day",
                main=paste(c("Temperature in", month.abb[month], year), collapse = " "),
                col.regions = colorRampPalette(c("deeppink", "blue", "cyan", "lightgreen", "yellow",
"orange", "red"))(length(breaks)-1),
                at= seq(-15, 35, length.out=120))
  plots = append(plots, list(x))
}
rm(year)
for (i in 1:12) {
  print(plots[i])
}
```

```
scatter3D(x=data.plot$Humidity, y=data.plot$Pressure, z=data.plot$Temp,
        clab = c("Temperature"),
        xlab="Humidity",
        ylab="Pressure",
        zlab="Temperature", col.panel ="white", pch = 18, bty = "u",
        col.grid = "black",
        main = "3D plot of Temp, Humidity, Pressure")

scatter3D(x=data.plot$Humidity, y=data.plot$Visibility, z=data.plot$Temp,
        clab = c("Temperature"),
        xlab="Humidity",
        ylab="Visibility",
        zlab="Temperature", col.panel ="white", pch = 18, bty = "u",
        col.grid = "black",
        main = "3D plot of Temp, Humidity, Visibility")

scatter3D(x=data$Processed.Summary, y=data$Precip.Type, z=data$Temp,
        clab = c("Temperature"),
        xlab="Processed.Summary",
        ylab="Precip.Type",
        zlab="Temperature", col.panel ="white", pch = 18, bty = "u",
        col.grid = "black",
        main = "3D plot of Temp, Processed.Summary,\n Precip.Type")

scatter3D(x=data.plot$Day, y=data.plot$Month, z=data.plot$Temp,
        clab = c("Temperature"),
        xlab="Day",
        ylab="Month",
        zlab="Temperature", col.panel ="white", pch = 18, bty = "u",
        col.grid = "black",
        main = "3D plot of Temp, Day, Month")

scatter3D(x=data.plot$Hour, y=data.plot$Day, z=data.plot$Temp,
        clab = c("Temperature"),
        xlab="Hour",
        ylab="Day",
        zlab="Temperature", col.panel ="white", pch = 18, bty = "u",
        col.grid = "black",
        main = "3D plot of Temp, Hour, Day")
```

```
# Plot to find optimal k value for knn models
k = c()
accuracy = c()
for (i in 1:20) {
  t = sample(nrow(data.discretized), floor(dim(data.discretized)[1]*0.01))
  training_data = data.discretized[t, ]
  test_data = data.discretized[-t, ]

  # Fit a KNN model with k=5
  attributes = c("Temp", "Humidity", "Pressure", "Precip.Type", "Processed.Summary",
"Visibility", "Day", "Month", "Hour")
  knn_model1 <- kNN(Temp~.,
                    train = training_data[attributes],
                    test = test_data[attributes],
                    k=i)

  # Prepare a confusion matrix to compute the error rate of the model on training
  # data
  conf_matrix = confusionMatrix(reference=test_data$Temp, data=knn_model1)

  # Append values to vector for later plot
  accuracy = append(accuracy, conf_matrix$overall[1])
  k = append(k, i)

  # Cleanup
  rm(conf_matrix)
}

# Plot the k against accuracy to find optimal k value for KNN model
plot(x=k, y=accuracy, xlab="k", ylab="Accuracy", type="l", main="Performance of KNN
model using different k values")
rm(k)
rm(accuracy)

# Frequency plot of discretized temperature
par(las=2)
barplot(count(data.discretized, 'Temp')$freq,
        names.arg = count(data.discretized, 'Temp')$Temp,
```

```
      horiz=FALSE,
      cex.names=0.8,
      col=c("deeppink","darkblue","deepskyblue","cyan","chartreuse","yellow","orange","red"))

#####################################################################
# KNN MODEL FITTING
#
# Instead of keeping this as a regression problem, we discretize
# the response variable & turn it into a classification problem.
# We match the temperatures into group of temperature range and
# apply KNN classification algorithm to fit the model
# and compute the accuracy of the model
#####################################################################

# Classification KNN Model 1
# Use 80% of the input data for training & 20% of data for testing
# Perform KNN algorithm with the optimal k obtained from the graph
t = sample(nrow(data.discretized), floor(dim(data.discretized)[1]*0.8))
training_data = data.discretized[t, ]
test_data = data.discretized[-t, ]

# Fit a KNN model with k=8
attributes = c("Temp", "Humidity", "Wind.Bearing.deg", "Pressure", "Precip.Type", "Month",
"Day", "Hour")
knn_model1 <- kNN(Temp~.,
            train = training_data[attributes],
            test = test_data[attributes],
            k=8)

# Prepare a confusion matrix to compute the error rate of the model on training
# data
conf_matrix_knn = confusionMatrix(reference=test_data$Temp, data=knn_model1)

#####################################################################
# Ensemble Model Fitting
#
# Instead of using only 1 model to predict the data, we combine
# multiple classification models to produce model with better
# performance
```

```
# Models used in the ensemble model include: KNN models, Random
# forest model
#########################################################################
attributes2 = c("Temp", "Humidity", "Wind.Bearing.deg", "Processed.Summary","Pressure",
"Precip.Type", "Month", "Day", "Hour")
rand_forest <- randomForest(Temp ~.,
                            method="class",
                            data=training_data[attributes2])

rand_forest_pred = predict(rand_forest, newdata = test_data[attributes2])
conf_matrix_ensemble = confusionMatrix(reference=test_data$Temp,
data=rand_forest_pred)

#########################################################################
# MORE PLOTTING
#########################################################################

# Level plot of the confusion matrices to show performance of
# 2 classification models through graphical method
confusion_plot = function(data, main, sub) {

  normalized_data = data
  for (i in 1:dim(data)[1]) {
    for (j in 1:dim(data)[2]) {
      normalized_data[i,j] = data[i,j] / sum(as.matrix(data)[,j])
    }
  }

  levelplot(normalized_data,
            xlab="Prediction",
            ylab="Reference",
            main=main,
            sub=sub,
            col.regions = colorRampPalette(c("red2", "red", "orange2", "orange", "goldenrod1",
"yellow", "greenyellow", "chartreuse", "green3"))(length(breaks)-1),
            at= seq(0, 1, length.out=120),
            panel=function(...) {
              arg <- list(...)
              panel.levelplot(...)
```

```
            panel.text(arg$x, arg$y, round(arg$z,1))})
}

confusion_plot(data=conf_matrix_knn$table,
               main="Confusion Matrix Plot of The KNN Model with K=8",
               sub=paste("Accuracy", conf_matrix_knn$overall[1]))

confusion_plot(data=conf_matrix_ensemble$table,
               main="Confusion Matrix Plot of The \n Random Forest Model",
               sub=paste("Accuracy", conf_matrix_ensemble$overall[1]))

# Plot accuracy of 2 models when training against different amount of data
performance_plot = function() {
  accuracy_knn = c()
  accuracy_rforest = c()
  proportion = c(0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8)
  for (p in proportion) {
    # Sample 40% of the rows in dataframe for training and the rest for testing
    # the model
    t = sample(nrow(data.discretized), floor(dim(data.discretized)[1]*p))
    training_data = data.discretized[t, ]
    test_data = data.discretized[-t, ]

    ###########################################################################
    # KNN Model Fitting
    ###########################################################################
    attributes = c("Temp", "Humidity", "Wind.Bearing.deg", "Pressure", "Precip.Type",
"Month", "Day", "Hour")
    knn_model1 <- kNN(Temp~.,
                      train = training_data[attributes],
                      test = test_data[attributes],
                      k=12)

    # Prepare a confusion matrix to compute the error rate of the model on training
    # data
    conf_matrix_knn = confusionMatrix(reference=test_data$Temp, data=knn_model1)

    ###########################################################################
    # Ensemble Model Fitting
```

```
  ######################################################################
  attributes2 = c("Temp", "Humidity", "Wind.Bearing.deg", "Processed.Summary","Pressure",
"Precip.Type", "Month", "Day", "Hour")
  rand_forest <- randomForest(Temp ~.,
                              method="class",
                              data=training_data[attributes2])

  rand_forest_pred = predict(rand_forest, newdata = test_data[attributes2])
  conf_matrix_ensemble = confusionMatrix(reference=test_data$Temp,
data=rand_forest_pred)

  accuracy_knn = c(accuracy_knn, conf_matrix_knn$overall[1])
  accuracy_rforest = c(accuracy_rforest, conf_matrix_ensemble$overall[1])
 }

 # Plot KNN line
 plot(x=proportion,
     y=accuracy_knn,
     xlab="Amount of training data",
     ylab="Accuracy",
     type="o", col="blue", pch=18, lty=1, ylim=c(0,1),
     main="Performance of classification models when \n training with different amount of
data")

 # Add random forest line
 points(proportion, accuracy_rforest, col="red", pch=19)
 lines(proportion, accuracy_rforest, col="red",lty=2)

 # Add a legend
 legend("bottomright", legend=c("Random Forest Model", "KNN Model"),
       col=c("red", "blue"), lty=2:1, cex=0.8,
       title="Models")
}
performance_plot()

# Variable selection plot
varImpPlot(rand_forest, main = "Variable Importance for Random Forest Model")

# KNN Model Performance plot with different k
```

```
## LOOCV cross validation
set.seed(0)
t = sample(nrow(data.discretized), floor(dim(data.discretized)[1]*0.5))
plot_data = data.discretized[t, ]
CV.kknn.interactions <- train.kknn(Temp~., data=plot_data[attributes],
                                   distance = 2, kmax = 30, kernel = "optimal")
rm(plot_data)
rm(t)

# plot MSE and elbow method to determine K
par(mfrow = c(1,1))
plot(CV.kknn.interactions, main = "Optimizing for K") #, xlab = "Numnber of Clusters",ylab =
"RSE")
number.clusters = 5
points(number.clusters, CV.kknn.interactions$MEAN.SQU[number.clusters], col = "red",
pch = 20)
```

Appendix C

This is what everyone in the group worked on.

Brady ran and interpreted the multiple linear regression section, including analyzing the full model, running it against reduced models and comparing it to the exhaustive search. Nam ran and interpreted the temperature prediction as a classification problem section, including analyzing and comparing the decision tree models and the random forest models. Eric ran and interpreted the simple linear regression, including performing a transformation and comparing it to the original simple linear regression model. Tim did the prediction interval and some linear regression that was not used in presentation. also set up slides for presentation, adjusted paper and fixed grammar mistakes.