**Introduction**

Weather forecast is important for people to decide their action for the next day. Among those technical terms in the forecasting, temperature and precipitation are the most important for the public. Nowadays weather forecasts, especially the precipitation forecast is conducted with the help of satellite images of clouds. This require knowledge in meteorology. So we wonder if we can predict whether it will rain tomorrow based on the basic weather record (like temperature, precipitation, wind), with the classification method we learned in class.

We choose Seattle for our study as it usually has half of time in a year raining, so the data are balance. In order to make better forecasting, feature engineering is needed. Also the data comes in form of daily report by each station. Combining the weather report each day from different stations is also a challenge for our study. The location of weather stations and relationship between variables are discussed in the section of data exploration. And we are going to use Random Forest, K-NN and SVM for classification. At last part we are going to use the models trained with Seattle data to predict precipitation in another city to see whether the models are generally usable.

**Random Forest**

Random Forest is an algorithm developed from decision tree. Trees that are grown very deep tend to learn highly irregular patterns: they overfit their training sets. Random forests are a way of averaging multiple deep decision trees, trained on different parts of the same training set, with the goal of reducing the variance. This comes at the expense of a small increase in the bias and some loss of interpretability, but generally greatly boosts the performance in the final model.

The major advantage of random forest is that it is robust against the collinearity, and we can rank the importance of variable based on out-of-bag error (Breiman L (2001). "Random Forests"). As it is easy to carry out and require no data assumption, we decided to first run it to get a better idea about which variable is more vital in weather forecast.

As shown in figure 1, the PRCP is the most important variable. This is intuitively correct as it is the most relative variable. Also we can see our feature engineering plays an important role. The TDIF (temperature difference) and other variables about wind are ranking from 2nd to 6th. This give us a sense of what variable should be used in the KNN, since KNN prefers a lower dimension input data.

In figure 2, the error rate become stable when tree number is larger than 100. As the red line is the training error rate of predicting non-precipitated days and the green line is the precipitated days. The model has a better ability in predicting non-precipitated days. This can be numerically confirmed from the confusion matrix in table 1. The type 1 error is significantly lower than type 2 error, which means model make more mistake in predicting rain days. The overall training errors are shown in table 1.

Although random forest has many advantages, it still can’t deal with the situation when data are not linear separable. So in that case we try kernel SVM for the next step.

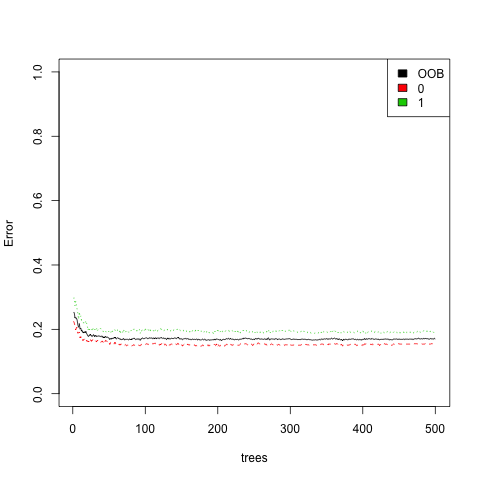
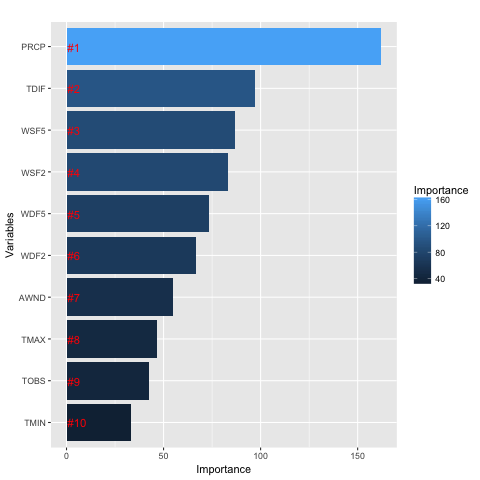


Figure 1: Importance of Attributes

Figure 2: Error Rate of Random Forest on Train Data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | True Condition | |  |
|  |  | Non-Rain | Rain |  |
| prediction | Non-Rain | 339 | 51 |  |
| Rain | 52 | 217 |  |
|  | Error Rate | 0.1329 | 0.1902 | 0.1562 |
|  |  | Type I | Type II | Overall |

Table 1: Confusion Matrix and Error Rate

**Distribution of Weather Stations**

The original data provided the longitude and latitude of the weather stations across Seattle. This is an important information as Seattle is surrounded by mountains and the weather records may vary a lot due to geographic differences.

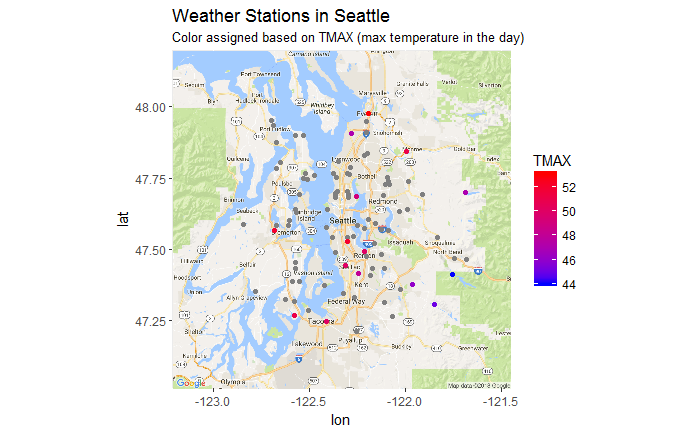
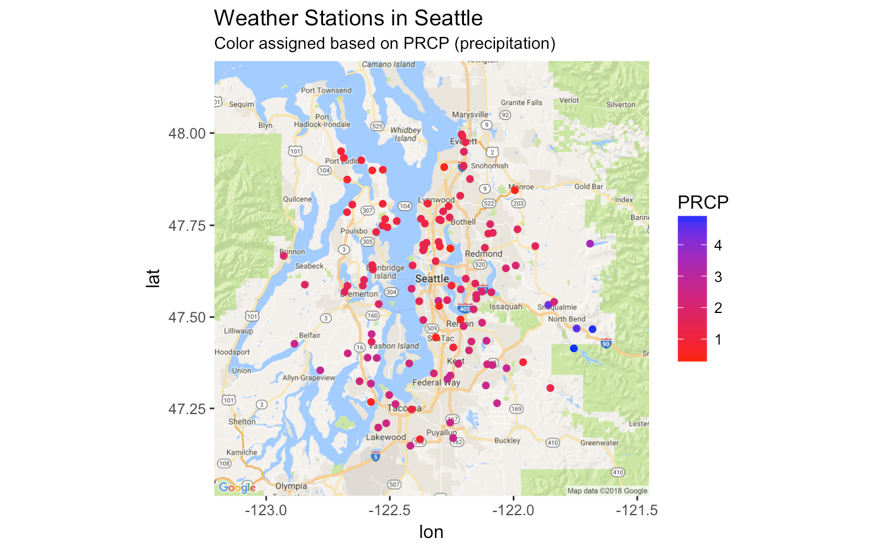


Figure 1: Station positions

The two map in Figure 1 is constructed with weather record in a particular day. The color of points in the map indicate the value of different variables, and the grey points means the value is not recorded (missing) in that day. As shown in left hand side, the south part of Seattle rained and the north part didn’t. Also there are 3 station at the east of Seattle city recorded high precipitation, which located in Mt Rainier. The max temperature records also vary a lot, between downtown Seattle and mountains around. So it is important to find a threshold of PRCP to decide whether it rained in Seattle that day.

Also to mention that the stations are randomly distributed across Seattle Area and shows no clustering pattern. Therefore, it is no point to perform clustering on stations’ location. Instead we compute the mean of weather records each day across all the stations. Most of the records are *scalar,* which can compute the mean directly. However, the records regarding wind, like wind directions and wind speed, are *vector,* so we computed the mean of vector sum of those records.

**Feature Engineer**

Despite the the variables provided by original data, we created some new variables to improve the model’s performance. As we are trying to forecast whether it rain in the next day, the temperature difference in each day could be important. The rain is formatted by condensation of water in air, higher temperature could contribute to evaporation of water and lower temperature may lead to condensation. Therefore, the temperature differences are useful in predicting raining.

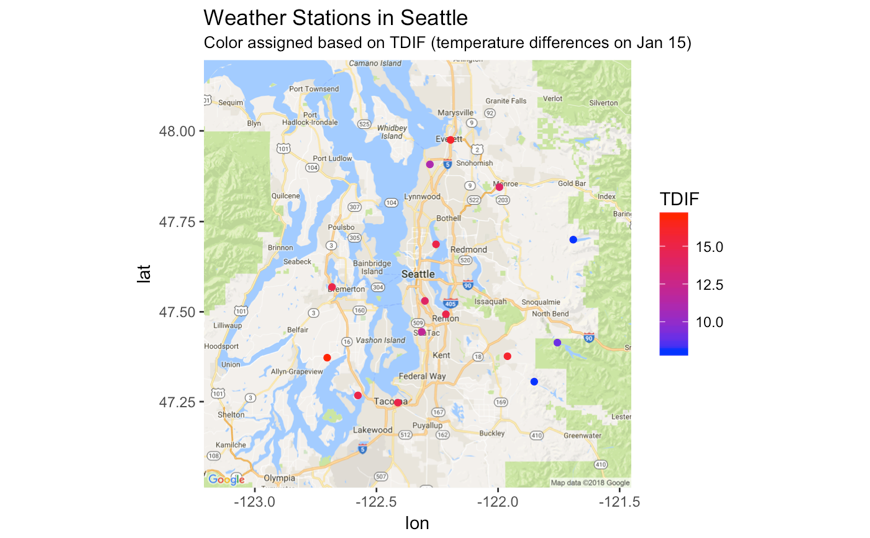
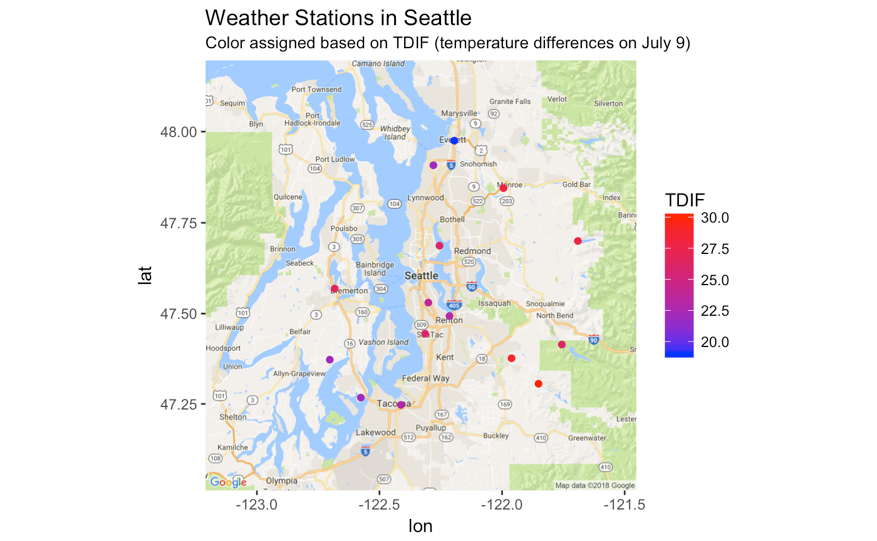


Figure 1: Stations with TDIF records

By comparing temperature differences (TDIF) in different days we find some interesting phenomenon. The overall TDIF is significantly larger in summer than in winter, which can be seen from the range of TDIF. And the TDIF at mountain areas is higher than downtown areas in summer, and opposite in winter. Those two phenomenon indicate that there could be a seasonal pattern in TDIF, and this could be the same for PRCP.

We also include a categorical variable as an indicator of whether it has precipitation at that day, based on the threshold of PRCP we find above.

**A short analysis of seasonal pattern in TDIF and PRCP**

Spectrum analysis, also referred to as frequency domain analysis, is the technical process of decomposing a complex signal into simpler parts. If we consider the time series of TDIF as a signal and if it has a seasonal pattern, we can use the spectrum analysis method to identify the dominant frequency.

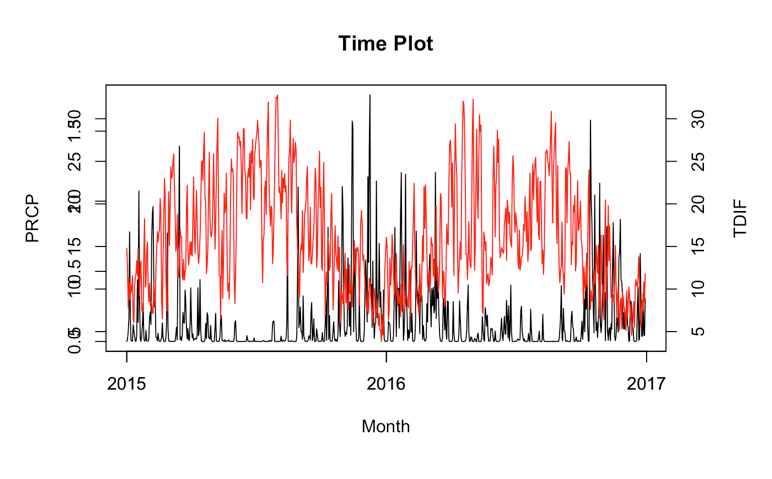
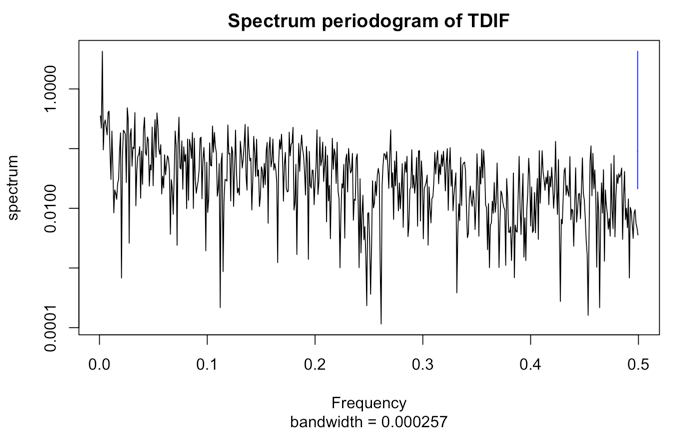


Figure 1: Spectrum Plot and Time Plot

First let’s see the plot on the left of Figure 1. There is an obvious peak in the plot, which is the frequency domain. I compute the frequency of that peak and the corresponding cycle, 375 days per cycle. This means the TDIF basically follow a cycle each 375 days. And there are no other peaks in the plot, so it indeed is the dominant one.

On the right hand side is the time plot of TDIF (red line) and PRCP (black line). The red line shows a clear cycle per year, this is corresponding to the spectrum analysis. On the other hand, the pattern of black line is not very clear. But it can tell us that PRCP is likely to be high when TDIF is low and vice versa. Overall even if we confirmed there is a seasonal pattern in PRCP (which should be true), it may not play a crucial role in predicting daily weather. However, what we learn from the analysis above is, the TDIF is important in predicting PRCP and they are likely to be not positively, but negatively correlated.

**Missing Value and Reason to Take Mean**

As we see from above, the attribute that recorded by each weather station are different, the day contains lots of missing values. The time sequence is significantly importance for our study so we can’t afford to remove any single row in the data. However, since we have data of weather stations’ record each day (approximate 100 stations each day), we can take mean of each attribute that is scalar (temperature, precipitation, etc), and take weighted vector sum of each vector attribute (attributes related to wind). By doing so we have a row that contains all the attributes for each date, and remove the attributes that still have missing value.

As we can see, the attributes recorded by each weather station are quite different, and there are lots of missing values for each day. However, since the time sequence is significantly importance for our study, we can not afford to remove any single row in the data. It is also not feasible to substitute the missing values because of the sparsity of the dataset. The first way is performing clustering based on the location of stations, substitute missing values with the mean of variables within each cluster. As shown in Figure 1, we have the largest average silhouette width when k = 2. The map plot on the right hand side of Figure 1 shows the clustering result, basically the stations are separated into two clusters, west and east, with downtown Seattle as the boundary in the middle. However, with this method we may create missing values since some of the variables are only recorded by stations in one cluster. Also it is hard to tell the relationship between clusters and the way to combine variables is harder. Therefore, we decided not to use clustering for substituting missing values.

Instead, since we have records from approximately 100 stations each day, we can group by date and take mean of each scalar attribute (temperature, precipitation, etc) directly, and take weighted mean vector of each vector attribute (attributes related to wind). By doing so we have a row that contains all the attributes for each date, and remove the attributes that still have missing values.

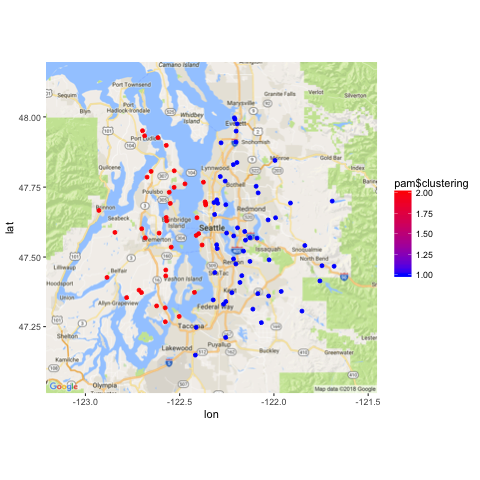
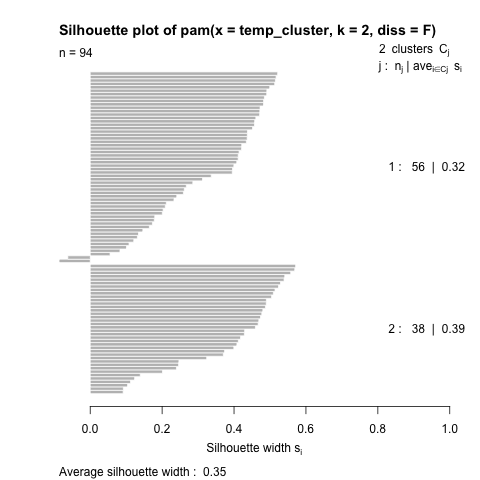


Figure 1: Clustering Results

**Using the model on Data from New York City**

So what about using the same model on the data from a different city? We consider to use the weather data from New York City in 2017 as test data on the model trained by Seattle data. We followed the same approach in data process as what we did on Seattle data. Since random forest and SVM have better performance than KNN, here we only test on those two models. The confusion matrix for random forest is shown in table 1 and in table 2 for SVM.

For random forest the Type I error is significantly higher. Although the Type II error is lower, it is actually due to the model classified to many days into rain days mistakenly. The model predicted 289 days that rain in 2017 and the average raining day in New York based on historical record is 122. Overall, this model trained with Seattle data isn’t suitable to predict precipitation in New York.

Again, we apply SVM with whole Seattle data as training set and test on New York. As shown in Table 2, the Type I error is significantly higher as well as the Type II error. And the Type II error is much less than the Type I error. Overall those two model performed badly in the New York data.

The reason this happened have several explanations. First, New York located on the east coast may have totally different meteorological environment compares to Seattle. So the variables and the threshold of those variables that matter in prediction may change. Second, New York has a rather imbalanced situation which has more days without precipitation, 238 non precipitation days and 126 precipitation days.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | True Condition | |  |
|  |  | Non-Rain | Rain |  |
| prediction | Non-Rain | 62 | 13 |  |
| Rain | 176 | 113 |  |
|  | Error Rate | 0.7394 | 0.1031 | 0.5192 |
|  |  | Type I | Type II | Overall |

Table 1: Confusion Matrix for Random Forest

**Confusion Matrix for New York Data in SVM**

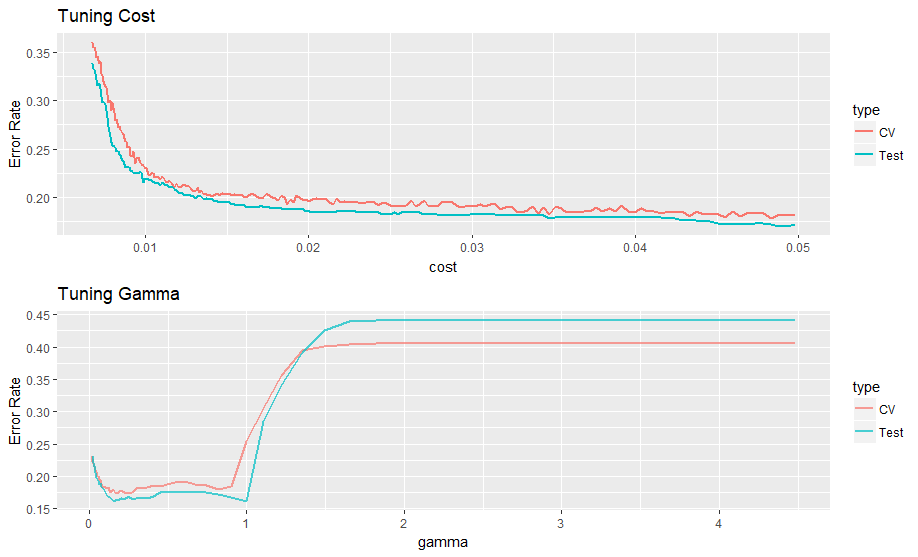
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | True condition | |  |
|  |  | Non-rain | Rain |  |
| Prediction | Non-rain | 96 | 45 |
| Rain | 142 | 81 |
|  | Error rate | 0.596 | 0.357 | 0.5137 |
|  | Type I | Type II | Overall |

Table 2: Confusion Matrix for SVM

**SVM (Modified by Hao)**

The effectiveness of SVM depends on the selection of kernel, the kernel's parameters, and soft margin parameter C. A common choice is a Gaussian kernel, which has a single parameter γ. The best combination of C and γ is often selected by a grid search with exponentially growing sequences of C and γ. Each combination of parameter choices is checked using cross validation, and the parameters with best cross-validation accuracy are picked. We can do this with function *tune* given a list of C. In figure 1 we can see the CV error and test error of SVM using different cost and gamma. The best C is 0.05 and best γ is 0.135.

With the parameters tunned we can conduct classfication. The test error are shown in Table 1. The Type I error is slightly larger than random forest, but Type II error is significantly lower than random forest. So SVM made less mistake in predicting raining days compare to random forest. This could partly be explained by the fact that this isn’t a linearly separable case.



**Confusion Matrix**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | True condition | |  |
|  |  | Non-rain | Rain |  |
| Prediction | Non-rain | 326 | 47 |
| Rain | 54 | 231 |
|  | Error rate | 0.142 | 0.169 | 0.1535 |
|  | Type I | Type II | Overall |