Seoul Bike Data Set:

Seoul Bike Data from UCI machine learning repository website.
This data contains 8760 instances, 14 attributes, and no missing values.

The categorical attributes are change to numerical attributes before use.

- Seasons - Winter (1), Spring (2), Summer (3), Autumn (4)
- Holidays - Holiday (1), No Holiday (2)
- Functional Day – Functional (1), Non Functional (2)

1.1 *Description of the classification task in broad practical terms.*

The target 3 classes (Low, Med, and High) are group by rented bike count. Balance the sizes of classes by changing the rented bike count cut off until each classes have roughly about the same size.

Classification		
Classes	**Conditions**	**Size**
Low	rented bike count < 260	2872
Med	260 ≤ rented bike count ≤ 850	2942
High	850 < rented bike count	2946
	Total	8760

1.2 *Detailed description of the data set.*

Currently Rental bikes are introduced in many urban cities for the enhancement of mobility comfort. It is important to make the rental bike available and accessible to the public at the right time. Eventually, providing the city with a stable supply of rental bikes becomes a major concern. The crucial part is the prediction of bike count required at each season for the stable supply of rental bikes.

SeoulBikeData contains 8760 instances, 14 attributes, and no missing values.

Attribute Information:

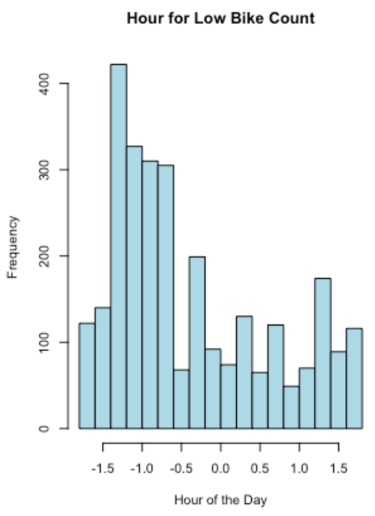
1. Date - year-month-day
2. Rented Bike count - Count of bikes rented at each hour
3. Hour - Hour of the day
4. Temp - Temperature in Celsius
5. Humidity - %
6. Windspeed - m/s
7. Visibility - 10m
8. Dew point - dew point temperature Celsius
9. Solar - radiation MJ/m2
10. Rainfall - mm
11. Snowfall - cm
12. Seasons - Winter, Spring, Summer, Autumn
13. Holiday - Holiday/ No holiday
14. Functional - Functional Hours/ Non Functional Hours

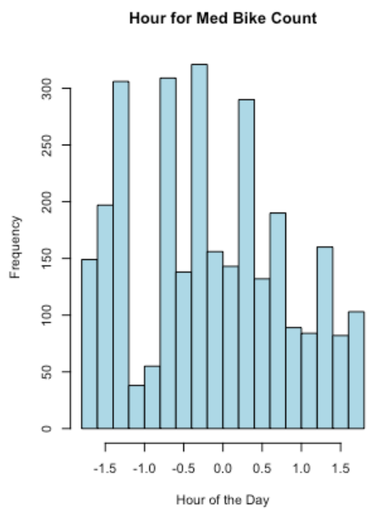
Discarded unneeded features date and rented bike count columns from the data set. We do not need rented bike count anymore since we already have class column with low, med, and high classes. Then we going to standardize each feature.

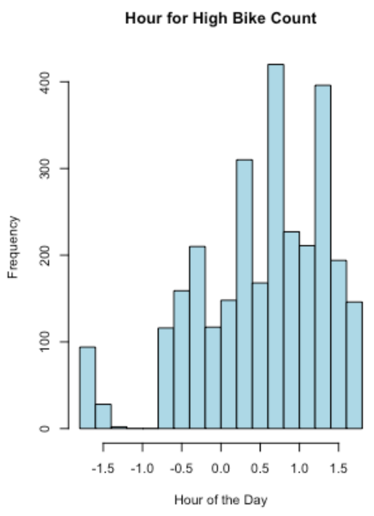
SeoulBike_clean contains 8760 instances and 13 attributes (including class column).

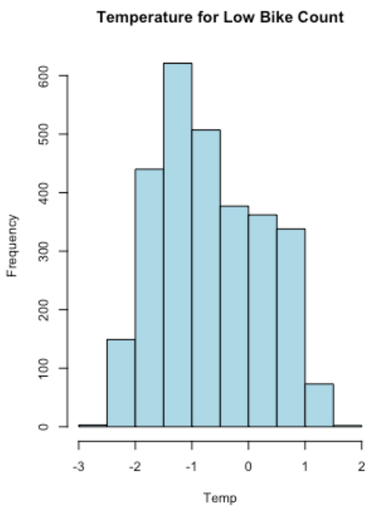
1.3 *Evaluate the discriminating power of a few features*

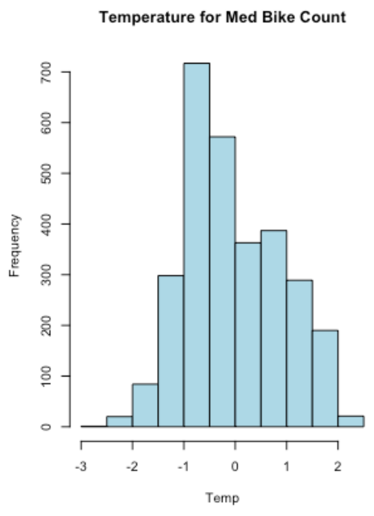
We going to create a standardize dataset SBike by standardizing the SeoulBike_clean dataset. Then SBike will be divided into 3 subsets by classes of LOW, MED, and HIGH. Using the new subset, we will create histograms of a few feature.

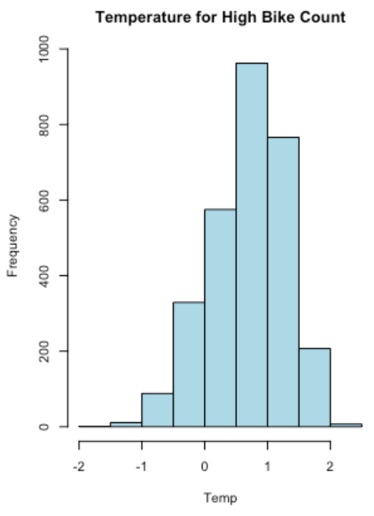


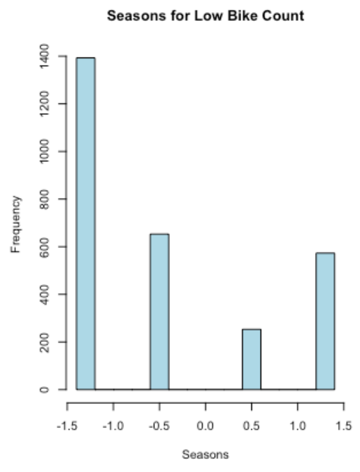


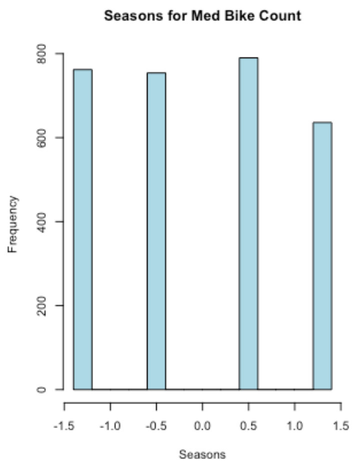


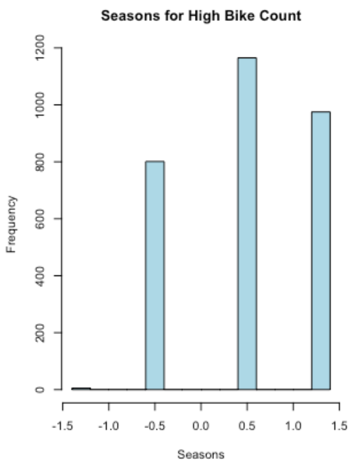












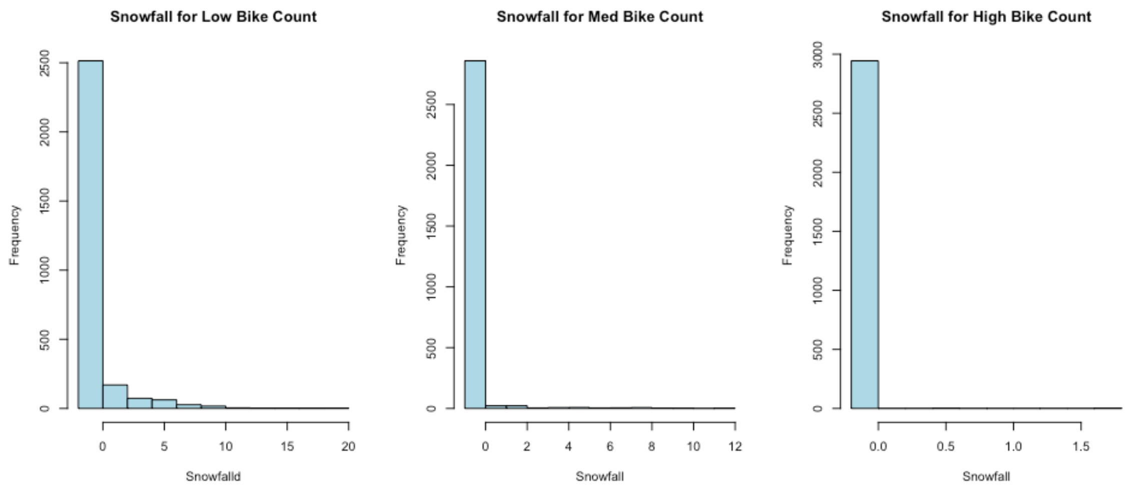


Figure 1: Histograms of 4 features (Hour of the Day, Temperature, Seasons, Snowfall) for different class
These histograms show the trends of each feature at different class.

Based on these histograms, we can see that hour, temperature, and seasons show right-skewed distribution for low bike count, symmetric distribution for med bike count, and left-skewed distribution for high bike count. The three features show trends that can be used to differentiate classes. On the other hand, snowfall did not show any different trends between the classes. Snowfall only shows right-skewed distribution for three classes. This means that we cannot use the rainfall feature to differentiate classes.

Features	Compare	D	P-Value
Temp	Low vs Med	0.30485	< 2.2e-16
	Low vs High	0.59266	< 2.2e-16
	Med vs High	0.4312	< 2.2e-16
Snowfall	Low vs Med	0.09644	3.637e-12
	Low vs High	0.12397	< 2.2e-16
	Med vs High	0.027533	0.2144
Hour	Low vs Med	0.21684	< 2.2e-16
	Low vs High	0.48576	< 2.2e-16
	Med vs High	0.32087	< 2.2e-16
Seasons	Low vs Med	0.22602	< 2.2e-16
	Low vs High	0.48333	< 2.2e-16
	Med vs High	0.25731	< 2.2e-16

Table 1: Results of KS test of 4 features (Hour of the Day, Temperature, Seasons, Snowfall) comparing Low vs Med, Low vs High, and Med vs High
The D statistic show the distance between the CDFs of the two samples.

Next, we perform the KS test using ks.test() function and SBike for each feature comparing the 3 classes (Low vs Med, Low vs High, and Med vs High*)*. Based on the KS test results, we can see that hour, temperature, and seasons show small p-value less than 2.2e-16 for all comparison of Low vs Med, Low vs High, and Med vs High. Snowfall have p-value less than 0.05 for Low vs Med and Low vs High. Since the P value is very small, then two groups were sampled from populations with different distributions. While Med vs High of snowfall have p-value higher than 0.05, then the groups were sampled from populations with the same distribution. D for hour, temperature, and seasons show that the two groups were sampled from population with different distributions. Since D for snowfall is close to 0, this show that the two groups were sampled from population with the same distribution. This feature that have groups from different distribution can be used to differentiate classes.

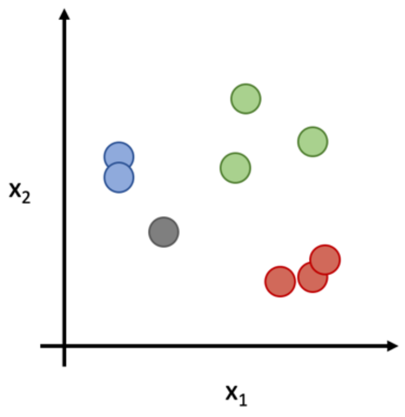
2.1 *Preparing a training set and a test set*

Taking the SBike set, we will separate each class into subsets of LOW, MED, and HIGH to create train and test sets. Each subset will then split randomly where train set will be about 80% of the subset and test set will be about 20% of the subset. After creating train and test sets for each class subset, all train sets will be unionized into a full Training_set (trainset_LOW, trainset_MED, trainset_HIGH). Then all test sets will be unionized into a full Test_set (testset_LOW, testset_MED, testset_HIGH).

3.1 *KNN (K-Nearest Neighbor) Algorithm Explanation*

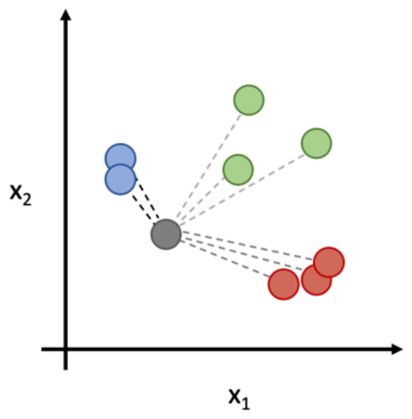
The KNN algorithm uses feature similarity to classify data. KNN finds the neighbors of an observation from using a measure distance such as Euclidean or Manhattan. Manhattan distance is better when there are many redundant features in the data. There are 4 steps to create a KNN algorithm:

1. Look at the data



In this example, you want to classify the grey point into a class. There are 3 potential classes – blue, green, and red.

2. Calculate distance between any two points



Start by calculating the distances between the grey point and all the other points.

3. Find nearest neighbors based on distances

Point	Distance	
⚫ - 🔵	2.1	1st NN
⚫ - 🔵	2.4	2nd NN
⚫ - 🟢	3.1	3rd NN
⚫ - 🔴	4.5	4th NN

Find the nearest neighbors by ranking points by increasing distance. The nearest neighbors (NNs) of the grey point are the one closet in dataspace.

4. Vote on a class based on nearest neighbor

Class	# of Votes
🔵	2
🟢	1
🔴	1

Vote on the predicted class labels based on the classes of the k nearest neighbors. In this example the labels were predicted based on the k = 3 nearest neighbors.

3.2 *Implement KNN classification for multiple values of K*

Using the knn() function, we will find percent accuracy for K = 5,10,15,20,30,40,50,60,70,80,90,and 100 for both train and test sets. The accuracy vs K plot will shows if the values will overfit between each set.

Train Set	
K	**% Accuracy**
5	85.70
10	81.99
15	80.29
20	79.20
30	77.68
40	76.26
50	75.78
60	75.14
70	74.75
80	74.41
90	73.98
100	73.94

Test Set	
K	**% Accuracy**
5	77.99
10	78.11
15	77.08
20	77.48
30	76.68
40	76.34
50	75.77
60	75.37
70	75.20
80	74.69
90	74.34
100	74.12

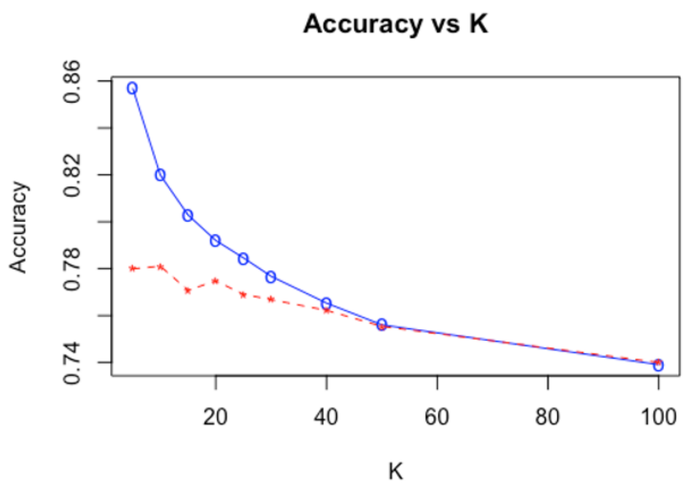


Figure 2: Plot of percent accuracy with varying k with both train and test sets.
A plot showing percent accuracy which is obtained by using k = 5,10, 15, 20,30,40,50, 100. The blue line is the train set and the red line is the test set values.

According to the accuracy vs K plot, the discrepancy between the curves show the magnitude of overfit. The table show 5 as ideal k value for best accuracy.

3.3 *Compute 3x3 confusion matrices for KNN using K = 5*

To use the KNN algorithm, a matrix with the predictors in the train set is created labeled TRAIN_Bike_no. Similarly, a matrix containing the predictors in the test set is created labeled TEST_Bike_no. Then two vectors are created that contain the class labels for the training observations and testing observations which are labeled TRAIN_Bike_label and TEST_Bike_label respectively. Using the train set and test set, we will apply the knn() function on both set to predict the class classification. We will run KNN algorithm on train and test sets using best K = 5 to find the accuracy values. In order to visualize the result, we will create a 3x3 confusion matrix.

		TRAIN PREDICTION		
	Test K = 5	**LOW**	**MED**	**HIGH**
TRUE	**LOW**	89.78%	0.76%	9.46%
	MED	1.22%	86.16%	12.62%
	HIGH	8.89%	10.18%	80.93%

		TEST PREDICTION		
	Test K = 5	**LOW**	**MED**	**HIGH**
TRUE	**LOW**	82.15%	1.18%	16.67%
	MED	2.36%	81.65%	15.99%
	HIGH	15.55%	14.66%	69.79%

Figure 3: Confusion matrices and percentage accuracy of TESTSET and TRAINSET
The left table is the accuracy percentage for train and the right table is for test set. They are showing predictions in rows and actual values in columns.

Train set

Based on the figure, 89.78% of "LOW" are classified correctly as "LOW", 86.16% of "MED" are classified correctly as "MED", 80.93% of "HIGH" are classified correctly as "HIGH". 10.22% of "LOW" were classified incorrectly as "MED" or "HIGH". 13.84% of "MED" were classified incorrectly as "LOW" or "HIGH". 19.07% of "HIGH" were classified incorrectly as "LOW" or "MED". LOW have the highest accuracy of 89.78% compare to MED of 86.16% and HIGH of 80.93%.

Test set
Based on the figure, 82.15% of "LOW" are classified correctly as "LOW", 81.65% of "MED" are classified correctly as "MED", 69.79% of "HIGH" are classified correctly as "HIGH". 17.85% of "LOW" were classified incorrectly as "MED" or "HIGH". 18.35% of "MED" were classified incorrectly as "LOW" or "HIGH". 30.21% of "HIGH" were classified incorrectly as "LOW" or "MED". LOW have the highest accuracy of 82.15% compare to MED of 81.65% and HIGH of 69.793%.

In conclusion, "LOW" had the best prediction with K value of 5.

Percent correct classifications:

Train accuracy = 85.70%
Test accuracy = 77.99%

The accuracy of test set is lower than the accuracy of train set. This is expected because the model selected for its accuracy on the training dataset rather than its accuracy on an unseen test dataset is likely to have lower accuracy on an unseen test dataset. The train accuracy will decrease (increase bias) with increase in K, but the test accuracy may increase at the same time (decrease variance). The model become more complex with higher K since it has to consider more neighbors. More complex model means that it capture the data distribution better to fit the train set perfectly, in other word, overfitting.

4.1 *PCA (Principal Component Analysis)*

PCA is used as a dimensionality-reduction method to explain the variance-covariance structure of a set of variables through linear combinations. The steps for PCA are:

1. Standardize the data and calculate the correlation matrix
2. Calculate the eigenvalues and eigenvectors of the correlation matrix
3. Examine and interpret the features contributions

4.1 *Correlation matrix heatmap*

We will standardize SBike data set and find the correlation of the set. Then we will create a heat map of the correlation matrix.

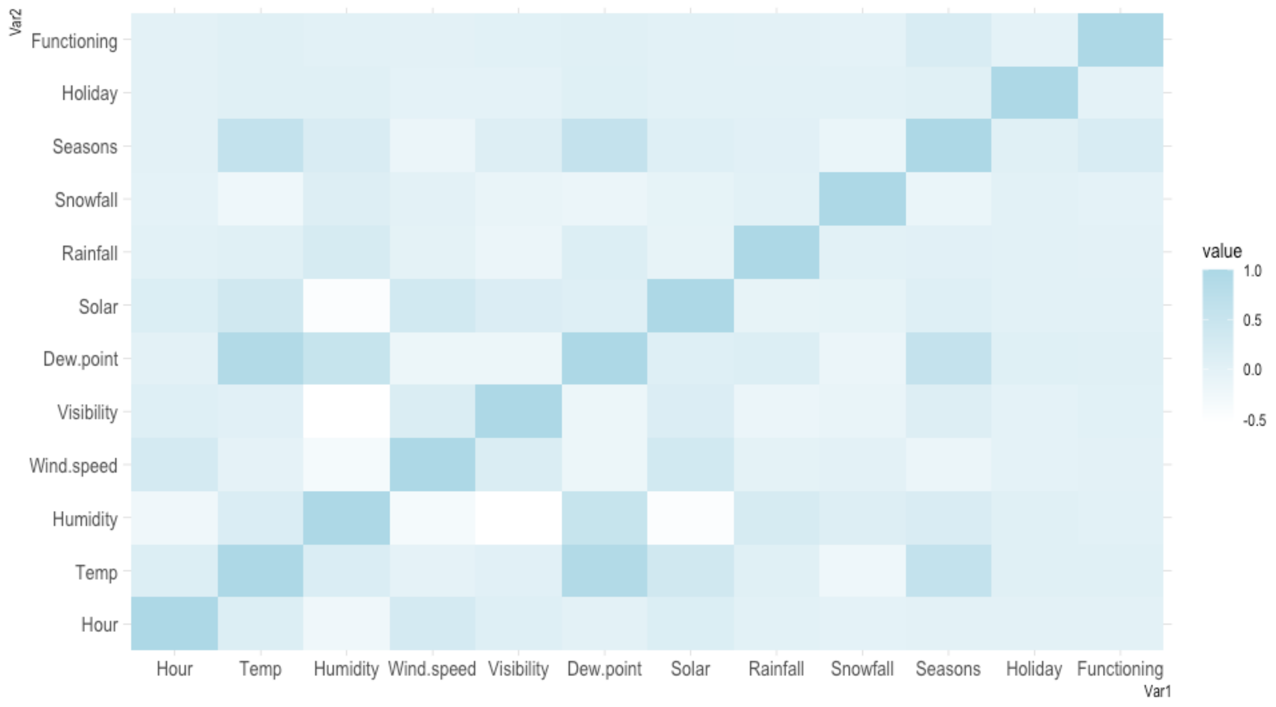


Figure 4: Correlation matrix heatmap
Blue means positive correlation and white means negative correlation. The darker it is the higher positive correlation. The whiter it is the higher negative correlation.

The matrix shows how strong the variables relate to each other. The closer it is to plus/minus 1, the stronger the relationship is. Positive correlation means one variable increases as other also increases. Negative correlation means one variable increases as the other decreases. The diagonal is 1 (dark blue) because those squares are correlating each variable to itself.

4.2 *Visualize eigenvalues (Scree Plot)*

Using prcomp() function to compute PCA, we can use the result to create the scree plot with fviz_eig() function. Scree plot is the plot of eigenvalues ordered from largest to smallest. It shows how much variation each principal component captures from the data. The plot is use to determine the principal components to keep in PCA.

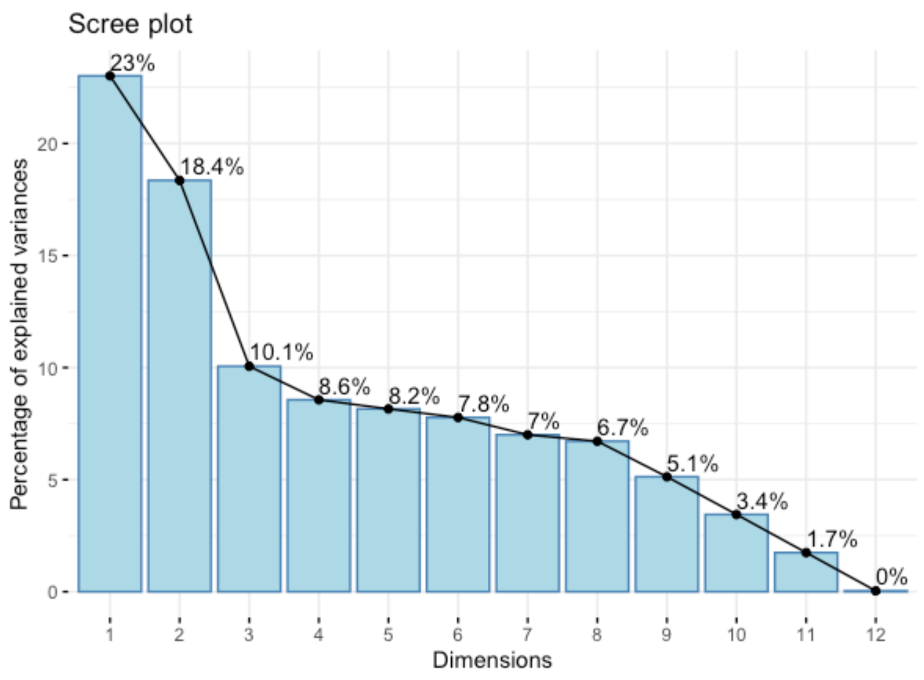


Figure 5: Scree plot
Principal component 1 (PC1) captures the most variation, PC2 captures the second most, and so on. Each of them contributes some information of the data.

According to the scree plot, about 95% of the information (variances) contained in the data are retained by the first 9 principal components.

4.3 *Visualize the contribution to first 9 principal components*

The contributions of variables in accounting for the variability in a given principal component are expressed in percentage. Variables that are correlated with PC1 – PC9 are the most important in explaining the variability in the data set because it contain 95% of the information. We going to use fviz_contrib() function to create a bar plot of variable contributions.

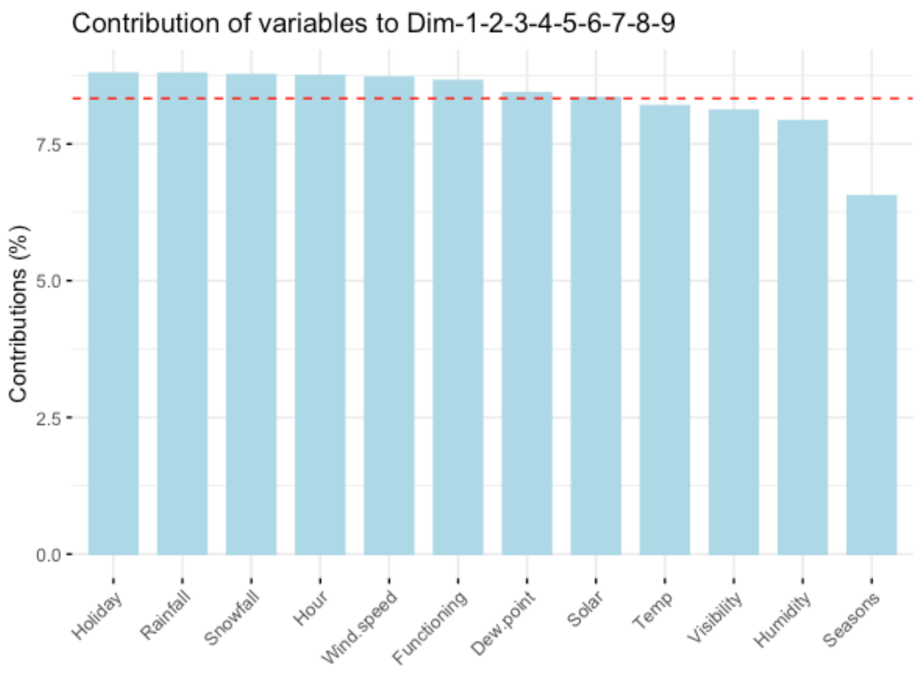


Figure 6: Contribution of individuals to the first nine principal components
Larger contribution percentage means more variable contributes to the component. Red dashed line on the graph above indicates the expected avg contribution.

According to the contribution plot, we can see that the variables – holiday, rainfall, snowfall, hour, wind speed, functioning day, and dew point contribute the most to the principal component 1-9 for 95% of the information. We will discarded the other 5 features solar, temperature, visibility, humidity, and seasons that made up the other 5% because of their low contributions.

4.4 *New subset based on PCA contribution results*

We going to take Training_set and Test_set and discard the other 5 features solar, temperature, visibility, humidity, and seasons. The new sets will now have 7 features. Then we will take the new Training_set_new and Test_set_new to create train and test labels.

4.5 *Compute 3x3 confusion matrices for KNN using the new train and test sets*

Using the Training_set_new and Test_set_new, we will apply the knn() function on both set to predict the class classification. We will run KNN algorithm on train and test sets using best K = 5 to create 3x3 confusion matrix.

		TRAIN PREDICTION		
	Test K = 5	**LOW**	**MED**	**HIGH**
TRUE	**LOW**	85.94%	4.31%	15.20%
	MED	1.22%	79.31%	10.18%
	HIGH	12.92%	13.67%	77.48%

		TEST PREDICTION		
	Test K = 5	**LOW**	**MED**	**HIGH**
TRUE	**LOW**	78.62%	4.55%	25.93%
	MED	1.01%	74.92%	13.30%
	HIGH	20.67%	18.20%	62.90%

Figure 7: Confusion matrices and percentage accuracy of the new TESTSET and TRAINSET
The left table is the accuracy percentage for train and the right table is for test set. They are showing predictions in rows and actual values in columns.

Train set
Based on the figure, 85.94% of "LOW" are classified correctly as "LOW", 79.31% of "MED" are classified correctly as "MED", 77.48% of "HIGH" are classified correctly as "HIGH". 19.51% of "LOW" were classified incorrectly as "MED" or "HIGH". 11.40% of "MED" were classified incorrectly as "LOW" or "HIGH". 26.59% of "HIGH" were classified incorrectly as "LOW" or "MED". LOW have the highest accuracy of 85.94% compare to MED of 79.31% and HIGH of 77.48%.

Test set
Based on the figure, 78.62% of "LOW" are classified correctly as "LOW", 74.92% of "MED" are classified correctly as "MED", 62.90% of "HIGH" are classified correctly as "HIGH". 30.48% of "LOW" were classified incorrectly as "MED" or "HIGH". 14.31% of "MED" were classified incorrectly as "LOW" or "HIGH". 38.87% of "HIGH" were classified incorrectly as "LOW" or "MED". LOW have the highest accuracy of 78.62% compare to MED of 74.92% and HIGH of 62.90%.

In conclusion, "LOW" had the best prediction with K value of 5.

Percent correct classifications

Train accuracy = 80.96%
Test accuracy = 72.29%

The accuracy of test set is lower than the accuracy of train set which means that there is overfitting.

Normal KNN Test Set

		PREDICTION		
	Test K = 5	LOW	MED	HIGH
TRUE	LOW	82.15%	1.18%	16.67%
	MED	2.36%	81.65%	15.99%
	HIGH	15.55%	14.66%	69.79%

KNN Test Set After PCA

		PREDICTION		
	Test K = 5	LOW	MED	HIGH
TRUE	LOW	78.62%	4.55%	25.93%
	MED	1.01%	74.92%	13.30%
	HIGH	20.67%	18.20%	62.90%

Figure 8: Confusion matrices and percentage accuracy of the original TESTSET and new TESTSET after PCA

Percent correct classifications

Normal KNN: Test accuracy = 77.99%
KNN after PCA: Test accuracy = 72.29%

The performance of the KNN algorithm decreases after taking out not important features according to the PCA. Usually, we expect to see an improvement in the performance after PCA. However, this decrease in performance is due to the fact that this data did not have many features (12 features) since the beginning. Then the features decreases even more to 7 features after PCA. The PCA treat the feature that has large variance as important features, but the feature with large variance can have nothing to do with the prediction target. For better future result, it is recommended to try LDA which takes response into account.