Predicting whether Lyft price of a ride is **cheap** or **expensive** based on weather using a decision tree.

In this particular model, I am going to use the weather dataset combined with the lyft-rides dataset that I got from Kaggle.

Within that dataset, I am going to use the weather dataset of Theater District (New York) on November 26, 2018 (**A rainy day**) as my training data.

**Decision Tree**

There are **5** weather elements that I am going to use as my **features**:

+Temperature (Farentheit)

+Cloud (Okta)

+Rain (mm)

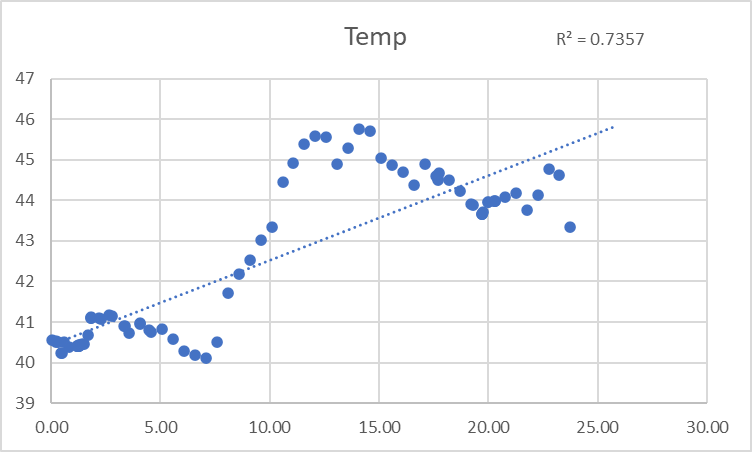
+Humidity (RH)

+Wind (mph)

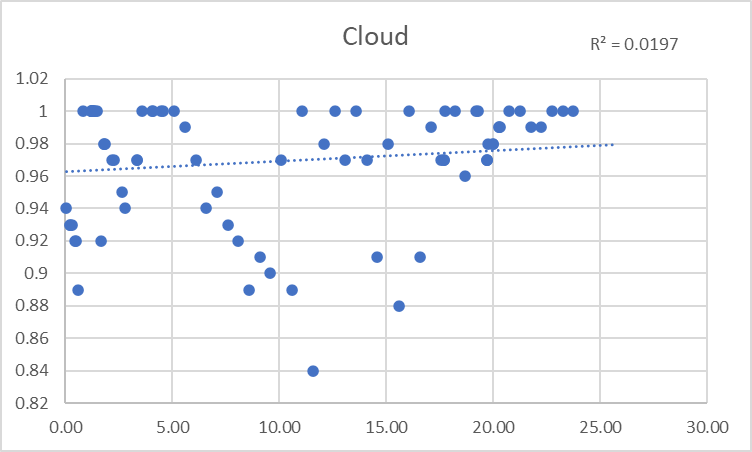
Theses are the visualization of each feature on November 26, 2018:

The x represents the time within the day and the y represents the measurement unit.

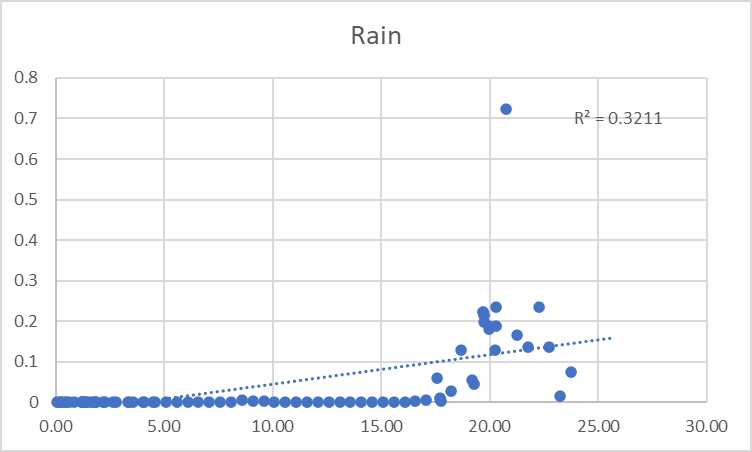
**TEMP**



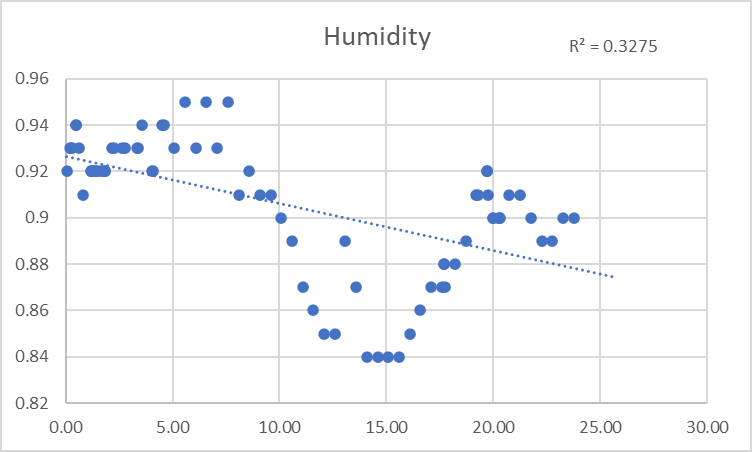
**CLOUD**



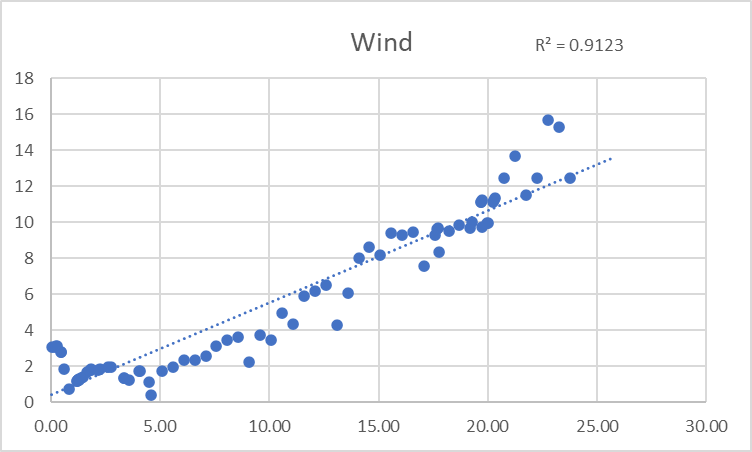
**RAIN**



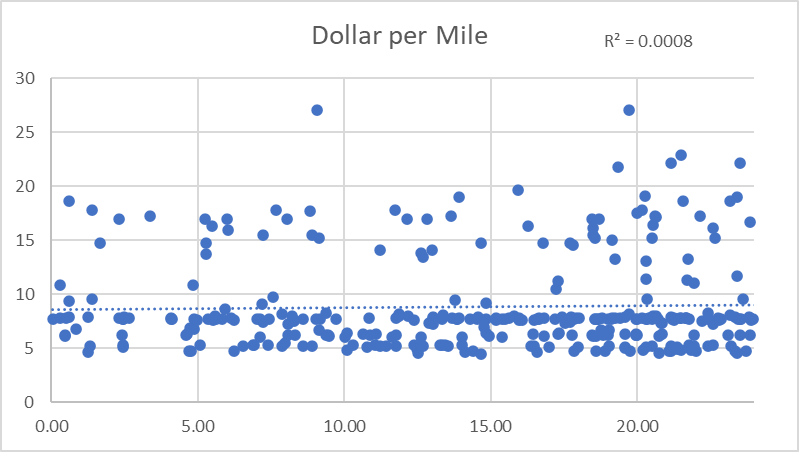
**HUMIDITY**



**WIND**



And here is the visualization of the average price per mile of every Lyft XL price within that day.



We can see that even though the weather features vary, R square of the price is consistent at around $8 Dollar a ride, which is the minimum fare of XL in 2018 according to Lyft.

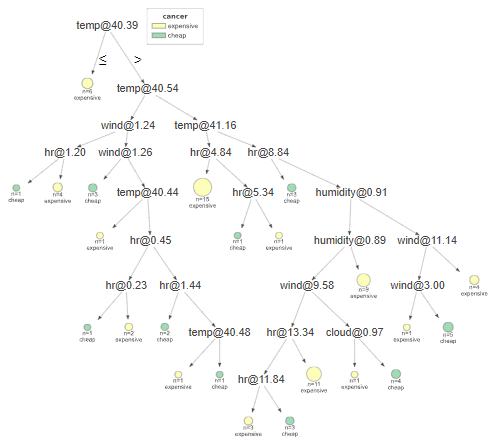
<https://www.lyft.com/pricing/BKN> (Inflation adjusted)

So then, for the sake of simplicity, we can classify that every ride that is above $8 is expensive.

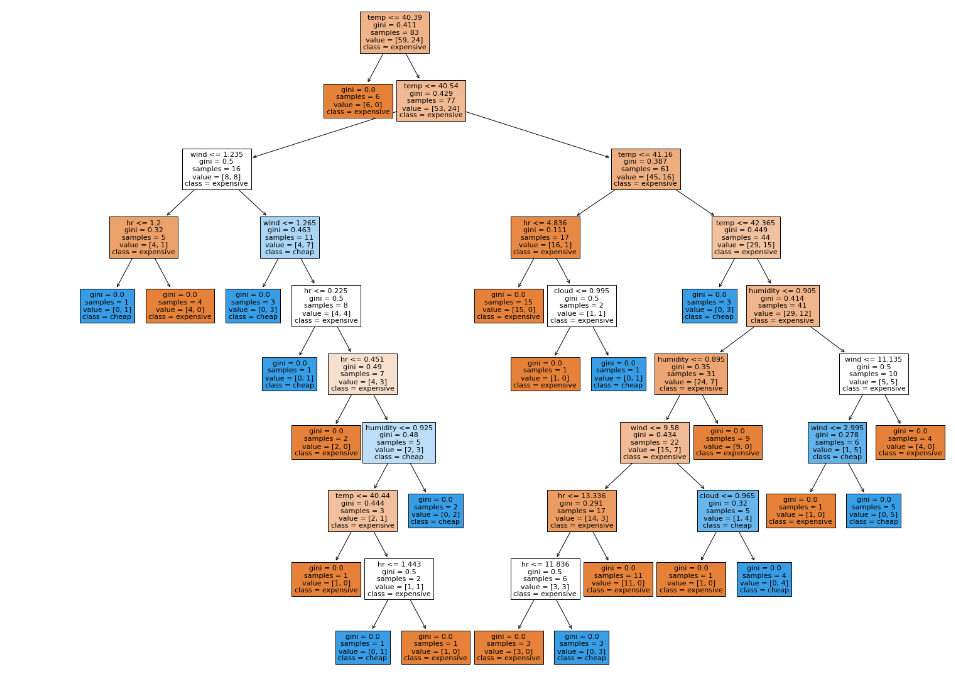
***So, when will the price be expensive?***

This is where I use a decision tree to determine whether the ride is going to be expensive based on the weather elements.

After using SKlearn Decision Classifier, this is the result decision tree:



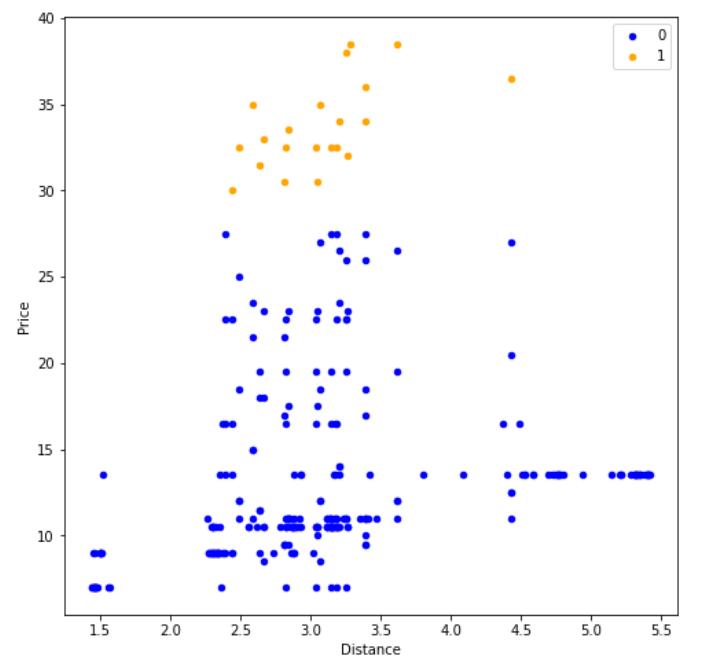
Another way to look at it with more detailed is:



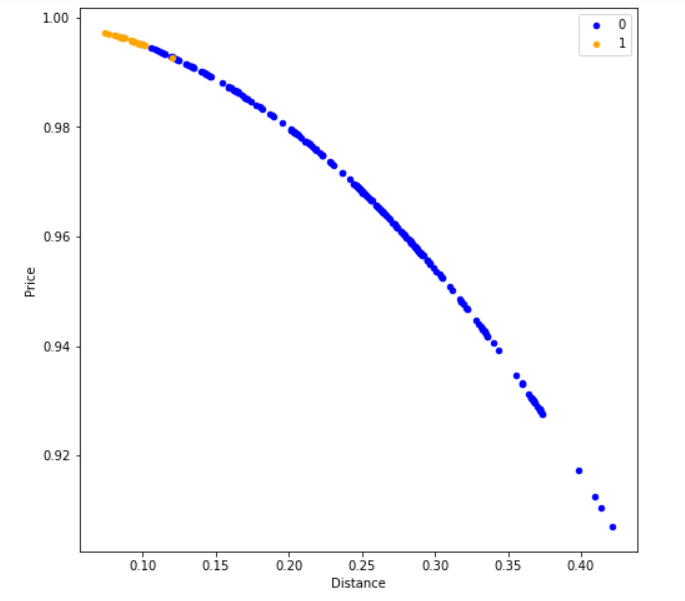
This decision tree is built without max\_depth and chooses to split at each height using **Gini index.**

**DBScan**

Compared to the previous method in which I used DBScan to classify whether a ride is cheap or expensive based on only 1 feature: **Distance**.



DBScan Clustering (After Removed “Noises”)



Data After Normalization.

The **yellow** dots represent where the price is expensive and **blue** dots represent where the price is cheap compared to the minimum fair in 2018.

We can see that, after normalizing our data and keeping the label the same, the price gets cheaper the further we go.

**Result**

We use a random dataset as train data and compared the result to the actual label

**Result using DBSCAN: 8 rights 5 wrongs**

**Result using Decision Tree: 5 rights 8 wrongs**