Predicting Beautiful Sunsets

Using Machine Learning Models and Social Media Data to Predict the Aesthetic Quality of a California Sunset

Jules Merigot

UCSB Fall 2022

Contents

Introduction **2**

Inspiration and Motive . . . 2

Data Description . . . 4

Project Outline . . . 4

Exploring Our Data **5**

Loading and Exploring the Data . . . 5

Tidying the Data . . . 6

Describing the Predictors . . . 8

Visual EDA . . . 9

Setting Up Models **18**

Train/Test Split . . . 18

Building Our Recipe . . . 19

K-Fold Cross Validation . . . 19

Building Prediction Models **20**

Performance Metric . . . 20

Model Building Process . . . 21

Model Results **21**

Visualizing Results . . . 21

Model Accuracies . . . 25

Results From Our Best Models **27**

Random Forest Model . . . 27

Support Vector Machine Model . . . 29

Putting the Model to the Test **31**

Beautiful Sunset Test 31

Bad Sunset Example 32

Extension: Final Model Accuracy **32**

Conclusion **33**

Introduction

The aim of this project is to build a machine learning model to predict how beautiful a sunset will be on any given day in the three biggest cities of the wonderful state of California. I will be using self-aggregated data pulled from a Stanford Computer Science Research Study, the official EPA website, and a weather history API source, all of which will be explained in further detail below along with their respective citations. Throughout this project, I will be implementing multiple machine learning techniques to yield the most accurate model for this binary classification problem.

Inspiration and Motive

Have you ever sat down and watched a sunset and thought to yourself, "Wow, that is stunning." Me too. Matter of fact, watching sunsets on the beach here in sunny Santa Barbara account for some of my fondest memories during my 4 years at UCSB. The rush of dopamine as you hurry down the street towards the beach, and the shot of serotonin when you run out onto the sand to discover the work of art that Mother Nature has prepared for you, is indescribable. You feel truly alive, and in my experience, when I watch a beautiful sunset, all of my problems seem to wash away in the massive sea that is the beauty of our planet.

However, with that beauty can also come the feeling of pure disappointment when you miss a once in a lifetime sunset. It's happened to me, it's happened to you, it's happened to all of us, but never again. In fact, I remember sitting on the deck of my house one day, admiring the sunset, and I thought to myself how sad it would be if I had missed it (seen below).

Then it occurred to me. What if we could predict when there would be a beautiful sunset? I turned the idea over in my head for a while. Most of us are aware of how difficult it is to predict weather already, so imagine predicting a sunset... But I am not one to give up because of something so trivial. I decided that if I was able to accumulate enough data combining past sunsets and their beauty scores, past air quality data, and meteorological data from those days, I would be able to predict the beauty of a sunset on any given day. While beauty is obviously subjective, the scores on which I will be basing my ratings and predictions on have been thoroughly analyzed, and will be explained later in the Data section. If not for me, then for all my friends, family, and fellow students, I want to be able to notify them of an incoming magical moment. The goal of this project is to help those with limited free time and an appreciation for the pureness of our environment to never miss a beautiful sunset again.

Data Description

I have myself combined this data into a csv file, binding on specific date and location. The completed dataset includes data from the following:

- "DETECTING AND PREDICTING BEAUTIFUL SUNSETS USING SOCIAL MEDIA DATA", a Stanford Computer Science Research Study by Emma Pierson.

A research paper conducted by a Stanford PhD student in 2016. For this study, 1.2 million sunset posts on Instagram, a picture-sharing platform, were collected and used to detect beautiful sunsets in 10 American cities over 7 months, from November of 2015 to May of 2016. For each city, 10 pictures were randomly selected from the five days with the highest amount of Instagram posts, and 10 pictures from days with a median amount of posts. These would be considered "beautiful" and "average" sunsets, respectively. Then, three research assistants were provided with one pair of sunset pictures at a time – one beautiful sunset, and one average sunset – in random order, and were tasked with choosing the one they felt had the more beautiful sunset. The three assistants agreed 86% of the time on average, indicating consensus about what constituted a beautiful sunset. Finally, a binary indicator variable for "beautiful sunsets" was created where a sunset with an average score in top 15% was denoted as a 1, and all others were denoted by 0. For more information on how these scores were drawn out and computed, visit the following hyperlink to the article: Stanford Sunset Data

- United States Environmental Protection Agency website.

The Environmental Protection Agency is an independent executive agency of the United States federal government tasked with environmental protection matters. This page contains large files of data intended for use by people who understand the EPA ambient air quality monitoring program and data. In particular, this website was used to collect hourly data from 2015 and 2016 on four criteria gases; carbon monoxide, ground-level ozone, nitrogen dioxide, and sulfur dioxide. I downloaded the CSV files and identified the specific date location needed to match my Instagram Sunset Data in order to paste the relevant observations into my large dataset which will be displayed later in this project. The website page with data used in this project can be found here: EPA Data

- Visual Crossing Weather Data & Weather API website.

A provider of weather data and enterprise analysis tools to data scientists, business analysts, professionals, and academics. I used Visual Crossing's Weather Data tool to collect meteorological data from specific time frames for the specific locations that I needed to match the Instagram sunset score data. I used the coordinates from three major cities, Los Angeles, San Francisco, and San Diego, and inputed their respective latitudes and longitudes to obtain downloadable CSV files with all the data observations I needed. I then copied this data and added it to my existing dataset, making sure to collate on specific date and location. For more information on Visual Crossing and for those interested in using their tools for their own projects, the website can be found here: Visual Crossing Data

Project Outline

Now that we have a better idea of the background and importance of our data, let's dive into our plans for our model in this project. To build our final binary classification model, we will follow a planned workflow to give us the best understanding of our process. First, we will clean our data by removing any large groups of missing observations, as well as remove any predictor variables that are unnecessary or complicating. With our remaining variables, we will perform further exploration tasks to gain a better understanding of how they are related to the aesthetic quality of sunsets in California. These variables will be used to predict a binary response variable named "Good Sunset", which will detail whether or not a sunset on a given day

was beautiful. We will then perform a training/test split on our data, make a recipe, and set folds for the 10-fold cross validation we will implement. Logistic Regression, Linear Discriminant Analysis, Quadratic Discriminant Analysis, Lasso, Decision Tree, Random Forest, and Support Vector Machine models will be all used to model the training data when we finish the setup. Once we have the results from each model, we will select the one that performed the best and fit it to our test dataset to discover how effective our model really is at predicting sunset beauty. Let's begin!

Exploring Our Data

We will first load the data that I have collected and delete some unnecessary variables. Since I assembled this dataset myself, there will be a lot less cleaning tidying required than larger datasets that are downloaded directly from a data source.

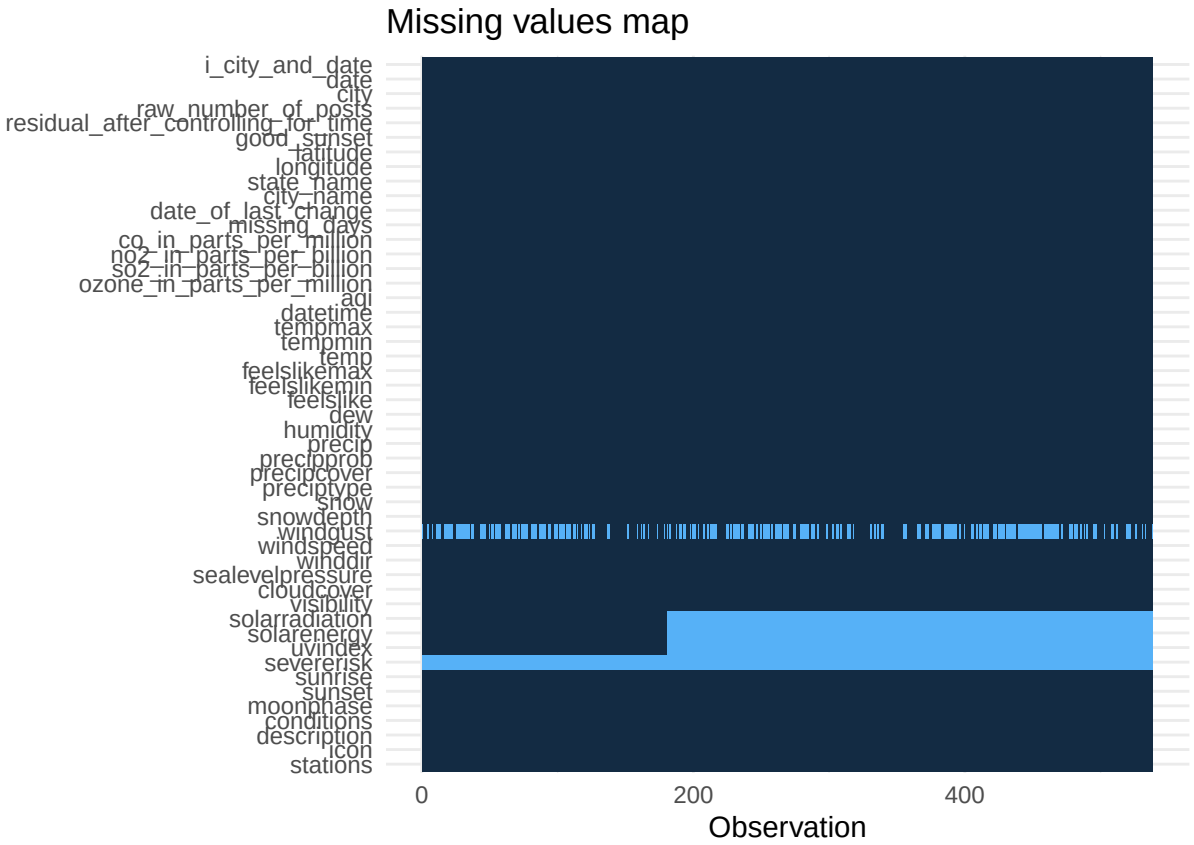
Loading and Exploring the Data

```
# loading the data
sunset_data <- read.csv("C:/Users/jules/OneDrive/Desktop/Sunset-Prediction-Project/Sunset-Data/sunset_q

# cleaning predictor names
sunset_data <- clean_names(sunset_data)
```

While this data was assembled by hand, there may still be some missing data. This happens often when collecting meteorological data if instruments are not working properly that day, or due to external factors not under our control. So let's check for missing data in our dataset. We'll make a missing plot to better visualize which variables are missing data points.

```
# plot of missing values in the data
sunset_data %>%
  missing_plot()
```



Total number of missing data observations:

```
# the number of missing values in the training data
sum(is.na(sunset_data))
```

```
## [1] 1891
```

Wow! As we can see in our missing values plot, we're missing a lot of data for `windgust`, `solarradiation`, `solarenergy`, `uvindex`, and `severerisk`. With so much missing data in all these predictors, we will be forced to remove them from the dataset entirely because it will otherwise impede on our predictions and cause errors.

Tidying the Data

Let's drop the variables with too much missing data. Additionally, some predictors that I originally kept will also not be needed so they will be removed below. We will also change our response variable `good_sunset`, which is currently a binary variable, to be a factor, and reorder the factor so that "Yes" is the second level. "Yes" will mean it was a good sunset, and "No" will be... well, its pretty self-explanatory. So let's do it!

```
drop <- c("i_city_and_date", "missing_days", "date_of_last_change", "datetime",
          "severerisk", "stations", "sunrise", "description", "city_name",
          "snow", "snowdepth", "windgust", "solarenergy", "solarradiation",
          "uvindex", "residual_after_controlling_for_time", "state_name",
          "latitude", "longitude", "raw_number_of_posts")
```

```
sunset_data = sunset_data[,!(names(sunset_data) %in% drop)]

# Make good_sunset a factor
sunset_data$good_sunset <- factor(sunset_data$good_sunset,
                                  labels = c("No", "Yes"))
```

One of our variables `date` which distinguishes the date of a particular sunset, is currently in mm/dd/yyyy format, which is difficult to use in a prediction or classification model, so let's extract the month and year from each value and make them their own predictors. The specific days won't matter as much here, but the particular month will have an impact since it traverses across seasons (November to May). It is commonly known that sunsets tend to be more vivid in color in the winter months, therefore there may be a significant difference in prediction results if a particular sunset's quality was evaluated in December of 2015 compared to one in May of 2016. While we're at it, let's also rename our air quality variables to something shorter so they take up less room in our visual plots later on. Don't worry, I'll explain what each one is in a bit.

```
dates <- as.POSIXct(sunset_data$date, format = "%m/%d/%Y")

sunset_data <- sunset_data %>%
  add_column("year" = c(format(dates, format="%Y")), .before = 1) %>%
  add_column("month" = c(format(dates, format="%m")), .before = 1) %>%
  plyr::rename(c("co_in_parts_per_million" = "co_in_ppm",
         "no2_in_parts_per_billion" = "no2_in_ppb",
         "so2_in_parts_per_billion" = "so2_in_ppb",
         "ozone_in_parts_per_million" = "ozone_in_ppm"))
```

Now that we've added these columns, let's make them numeric variables so they can be used in our recipe later on without being dummy coded.

```
# make Month and Year numeric variables
sunset_data$month <- as.numeric(as.character(sunset_data$month))
sunset_data$year <- as.numeric(as.character(sunset_data$year))
```

Below we can see the current dimensions of our dataset after removing certain variables and cleaning out the missing data values.

```
sunset_data %>% dim()
```

```
## [1] 538  31
```

Let's take quick look at our dataset by displaying the first 10 rows.

```
sunset_data %>% head()
```

```
##   month year      date        city good_sunset co_in_ppm no2_in_ppb so2_in_ppb
## 1    11 2015 11/1/2015 Los_Angeles          No       1.2       41.3        1.0
## 2    11 2015 11/2/2015 Los_Angeles         Yes       1.0       32.0        0.2
## 3    11 2015 11/3/2015 Los_Angeles         Yes       0.5       26.0        0.0
## 4    11 2015 11/4/2015 Los_Angeles          No       0.8       36.0        0.3
## 5    11 2015 11/5/2015 Los_Angeles          No       0.9       44.5        0.5
## 6    11 2015 11/6/2015 Los_Angeles          No       1.3       50.2        6.5
##   ozone_in_ppm aqi tempmax tempmin temp feelslikemax feelslikemin feelslike
```

```
## 1        0.062  39    83.9    59.7 69.9         81.3         59.7      69.7
## 2        0.034  30    73.4    59.9 65.7         73.4         59.9      65.7
## 3        0.035  25    67.7    53.0 60.0         67.7         53.0      60.0
## 4        0.033  34    67.3    51.3 59.1         67.3         51.3      59.1
## 5        0.034  42    69.9    50.3 60.1         69.9         50.3      60.1
## 6        0.030  47    76.1    51.6 63.1         76.1         51.6      63.1
##    dew humidity precip precipprob precipcover preciptype windspeed winddir
## 1 51.1     55.0   0.00          0        0.00                 7.4   222.3
## 2 55.6     70.3   0.01        100        4.17       rain     10.8   238.5
## 3 44.4     58.0   0.00          0        0.00                 9.8   280.0
## 4 35.9     44.6   0.00          0        0.00                13.9   286.2
## 5 29.9     32.9   0.00          0        0.00                 7.1   294.3
## 6 32.4     33.8   0.00          0        0.00                 6.3   246.9
##   sealevelpressure cloudcover visibility              sunset moonphase
## 1           1015.3        0.3        9.7 2015-11-01T17:00:07      0.65
## 2           1009.8       74.1        9.4 2015-11-02T16:59:11      0.70
## 3           1009.9       18.5        9.8 2015-11-03T16:58:16      0.75
## 4           1014.1        6.9        9.9 2015-11-04T16:57:22      0.80
## 5           1020.7        0.8        9.9 2015-11-05T16:56:30      0.85
## 6           1019.1        0.0        9.9 2015-11-06T16:55:40      0.89
##               conditions      icon
## 1                  Clear clear-day
## 2 Rain, Partially cloudy      rain
## 3                  Clear clear-day
## 4                  Clear clear-day
## 5                  Clear clear-day
## 6                  Clear clear-day
```

Describing the Predictors

After removing unnecessary variables, we can get a better understanding of what each predictor that we will using represents. The variables that I selected for the true data set and that I will be using in my model recipe to predict good sunsets later on are as following:

- `month`: The month that the sunset was recorded (ranges from November 2015 to May 2016) as represented by its respective number (e.g. 11 for November)
- `year`: The year that the sunset was recorded (either 2015 or 2016)
- `city`: The city in which the sunset was recorded (either Los Angeles, San Francisco, or San Diego)
- `co_in_ppm`: The carbon monoxide CO) levels in the air on the given day of the sunset, measured in parts per million
- `no2_in_ppm`: The nitrogen dioxide (NO2) levels in the air on the given day of the sunset, measured in parts per million
- `so2_in_ppm`: The sulfur dioxide (SO2) levels in the air on the given day of the sunset, measured in parts per million
- `ozone_in_ppm`: The ozone (O3) levels in the air on the given day of the sunset, measured in parts per million
- `aqi`: The air quality index on the given day of the sunset in the city it took place
- `tempmax`: The maximum recorded temperature on the given day of the sunset in the city in took place
- `tempmin`: The minimum recorded temperature on the given day of the sunset in the city in took place
- `temp`: The average recorded temperature on the given day of the sunset in the city in took place
- `dew`: The dew point is the temperature at which the air can exactly hold the amount of moisture present, in degrees Fahrenheit

- `humidity`: The humidity present in the air on the given day of the sunset, measured in g.m-3, which is units of grams of water vapor per cubic meter of air.
- `precip`: The precipitation on the given day, measured in inches of rainfall depth
- `precipprob`: The probability of precipitation on the given day of the sunset
- `precipcover`: The precipitation Coverage which is the percentage of time during the reporting window that the precipitation actually occurred
- `windspeed`: The wind speed on the given day, measured in miles per hour (mph)
- `winddir`: The wind direction on the given day, reported in degrees and describes the direction from which the wind emanates. A direction of 0 degrees is due North on a compass. A direction of 180 degrees is due South. A direction of 270 degrees would indicate a wind blowing in from the west.
- `sealevelpressure`: The sea-level pressure on the given day, measured in metric millibars
- `cloudcover`: The percentage of the sky that is covered by clouds on the given day
- `visibility`: Visibility on the given day, measured in miles of the greatest distance the average human eye can distinguish objects
- `moonphase`: The phase of the moon on the given day, where 1 represents a full moon, 0 represents a new moon, 0.5 represents a first or third quarter (half of the moon visible), and so on.

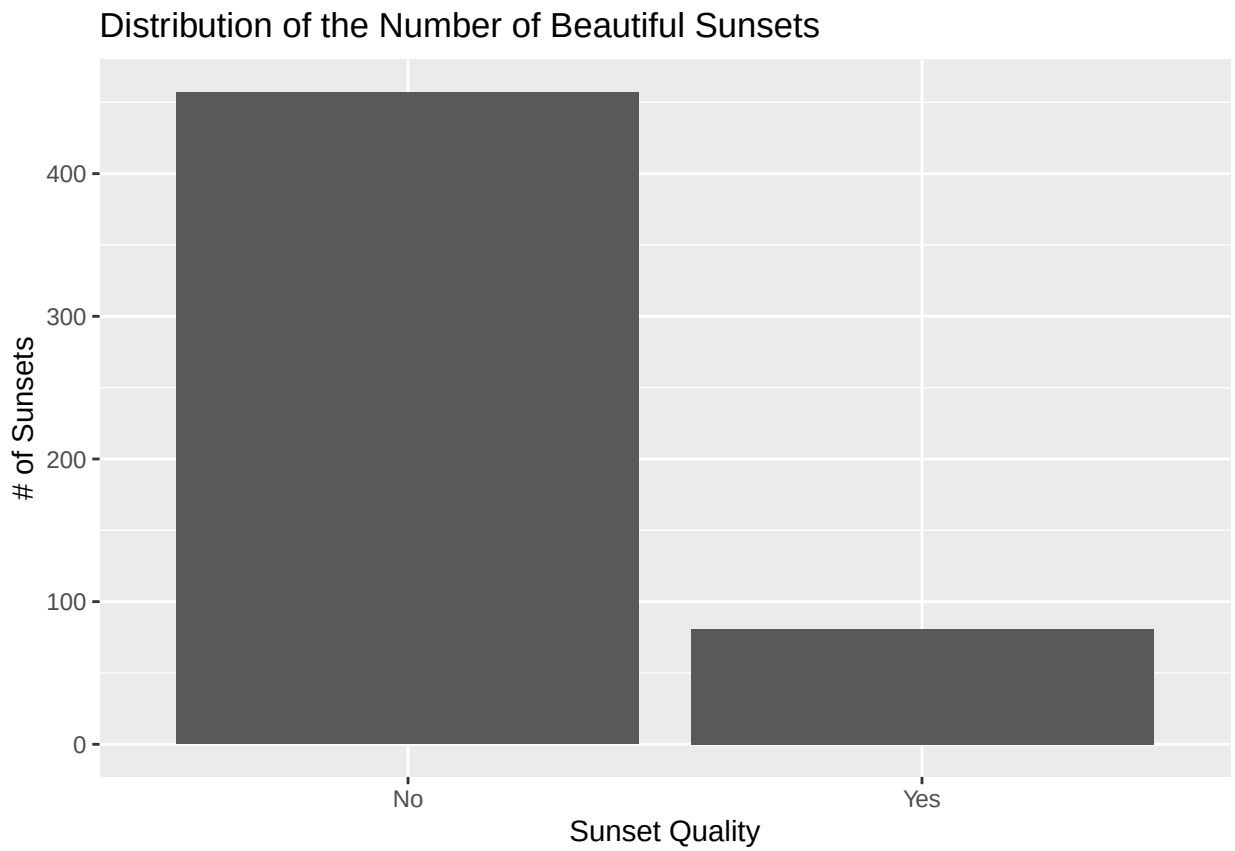
Visual EDA

To get a better idea of the particular distribution of our response variable and our predictors, we'll generate an output variable plot as well a correlation matrix to identify potential correlations between our predictor variables. Additionally, we'll create visualization plots to see the effect that certain variables of interest have on our response variable.

Good Sunset Distribution

One important thing to note before diving deeper into building our models, is to realize that beautiful sunsets are heavily outweighed in occurrence by poorer sunsets. This is expected of course, but can you imagine how awesome it would be if there was a stunning sunset every night!? I would never be able to get any work done! Anyways, below we can see a plot of the distribution of the sunsets and their respective aesthetic quality over the seven months. As a reminder, a "Yes" for a sunset means it was in the top 15% of the most beautiful recorded sunsets.

```
# looking at the distribution of the Good Sunsets
sunset_data %>%
  ggplot(aes(x = good_sunset)) +
  geom_bar() +
  labs(x = "Sunset Quality", y = "# of Sunsets", title = "Distribution of the Number of Beautiful Sunse
```

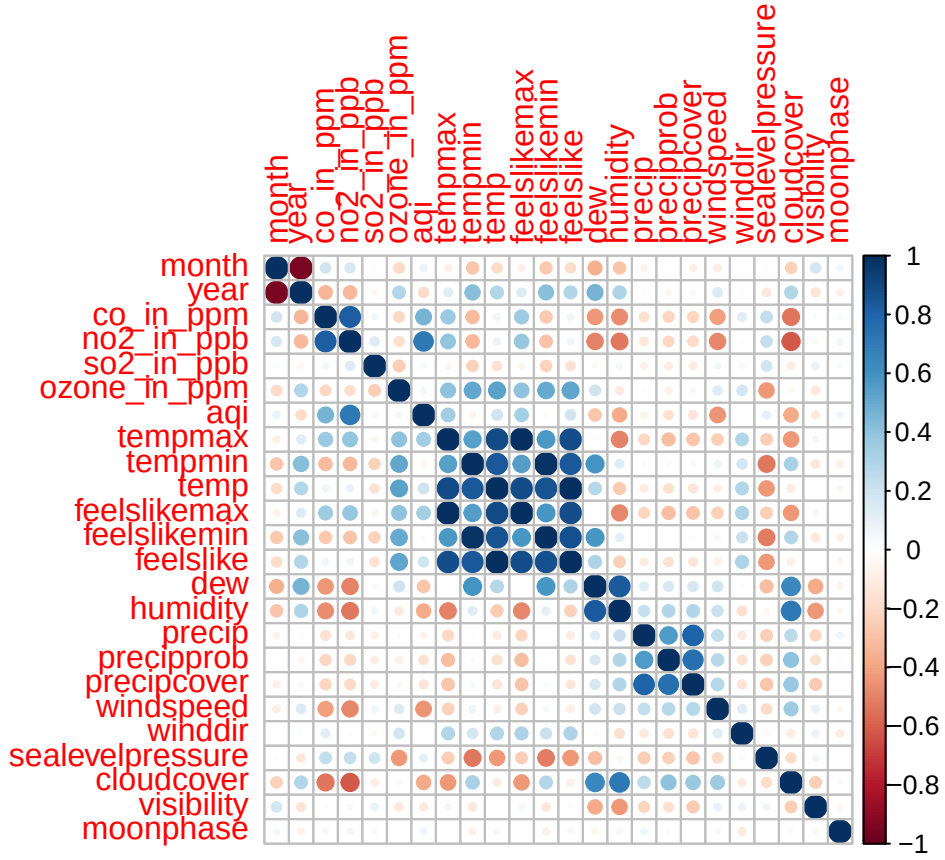


As we can see, while there were less than 100 sunsets that made the top 15%, an overwhelming amount of just slightly more than 450 sunsets were considered to be not as beautiful. This means that once we have built and fit our models, we'll be able to alert our friends of sunsets that are really worth it.

Variable Correlation Plot

To get an idea of the relationships between our numeric variables, we'll make a correlation matrix and then make a heat map of the correlation of these predictors.

```
# making a correlation matrix and heat map of the predictors
sunset_data %>%
  select(where(is.numeric)) %>%
  cor() %>%
  corrplot()
```



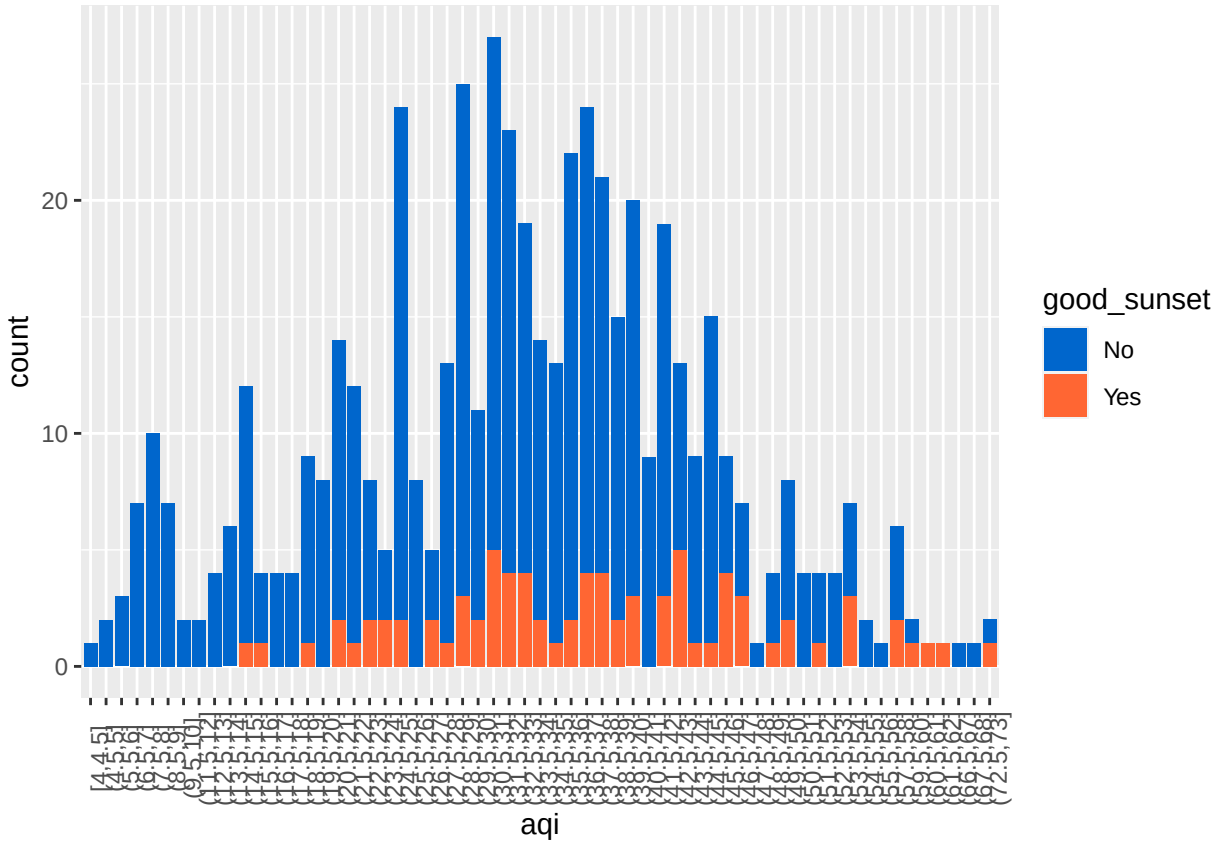
Wow those colors almost resemble those present during a sunset, how crazy. I was not too surprised to see so much correlation between my predictors since they encompass most weather and air quality data available, and therefore will tend to be related. However, a few stood out to me. First, the big blue box in the middle represent all the temperature variables which will obviously be strongly correlated in this instance. I was surprised to find how much of a positive correlation the ozone concentration in parts per million had with my various temperature variables. I was also interested to discover that the cloud cover of a given day was negatively correlated with the carbon monoxide and nitrogen dioxide levels in the air.

Air Quality Index (AQI)

Air Quality Index, also known as AQI, is a predictor of the quality of a sunset. In the case of the many forest fires that California has experienced over the past few years, the particles released by the burning of the forests produces an orange-ish color in the sky. While these particles and smoke and dangerous for living creatures to breathe, they also make for some beautiful and vivid orange sunsets. On the other hand, cleaner air also allows for less haze and therefore clearer sunsets, so both way work in favor of a good sunset. As we can see in the plot below, the distribution of good sunsets is present equally on the higher end and lower end of the AQI spectrum, but with more density of observations under an AQI of 40 (considered clean air). On the other hand, poorer sunsets are much more frequent when the AQI is lower, and thus when the air is cleaner.

```
sunset_data %>%
  dplyr::select("aqi", "good_sunset") %>%
  dplyr::mutate(aqi = cut(aqi, breaks =
                            seq(min(aqi), max(aqi), by = 0.5),
                          include.lowest = TRUE)) %>%
```

```
  group_by(aqi) %>%
  na.omit(aqi) %>%
  ggplot(aes(aqi)) +
  geom_bar(aes(fill = good_sunset)) +
  scale_fill_manual(values = c("#0066CC", "#FF6633")) +
  theme(axis.text.x = element_text(angle = 90))
```

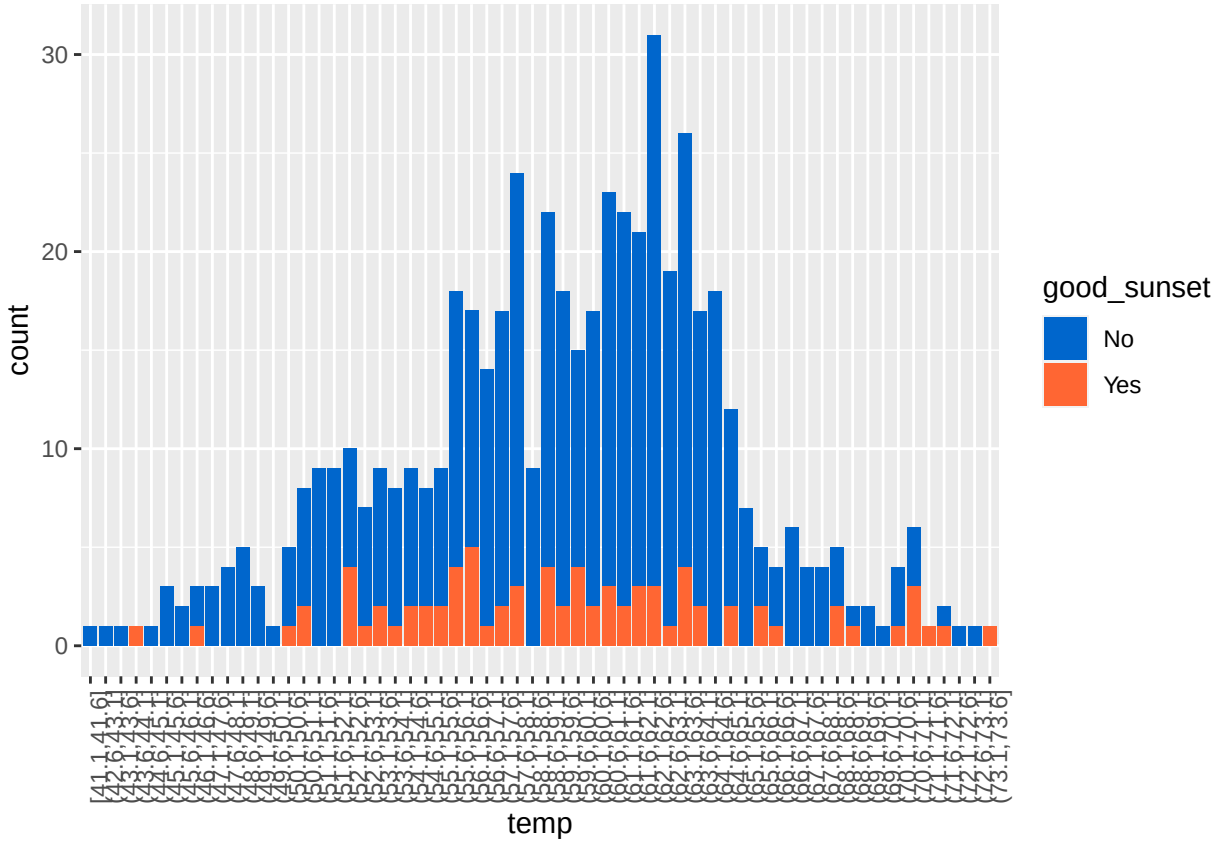


Temperature

Temperature can also significantly affect the quality of a sunset. As discussed earlier, winter sunsets tend to be more vivid during the winter. However, these more beautiful sunsets are usually due to less humidity and cleaner air due to rain and snow. As we can see below, good sunsets seem to be more frequent when temperatures are lower, below 60 degrees Fahrenheit. For those that are not aware, below 60 is considered cold in California, but one can imagine how these may look when it gets even colder...

```
sunset_data %>%
  dplyr::select("temp", "good_sunset") %>%
  dplyr::mutate(temp = cut(temp, breaks =
                             seq(min(temp), max(temp), by = 0.5),
                           include.lowest = TRUE)) %>%
  group_by(temp) %>%
  na.omit(temp) %>%
  ggplot(aes(temp)) +
  geom_bar(aes(fill = good_sunset)) +
```

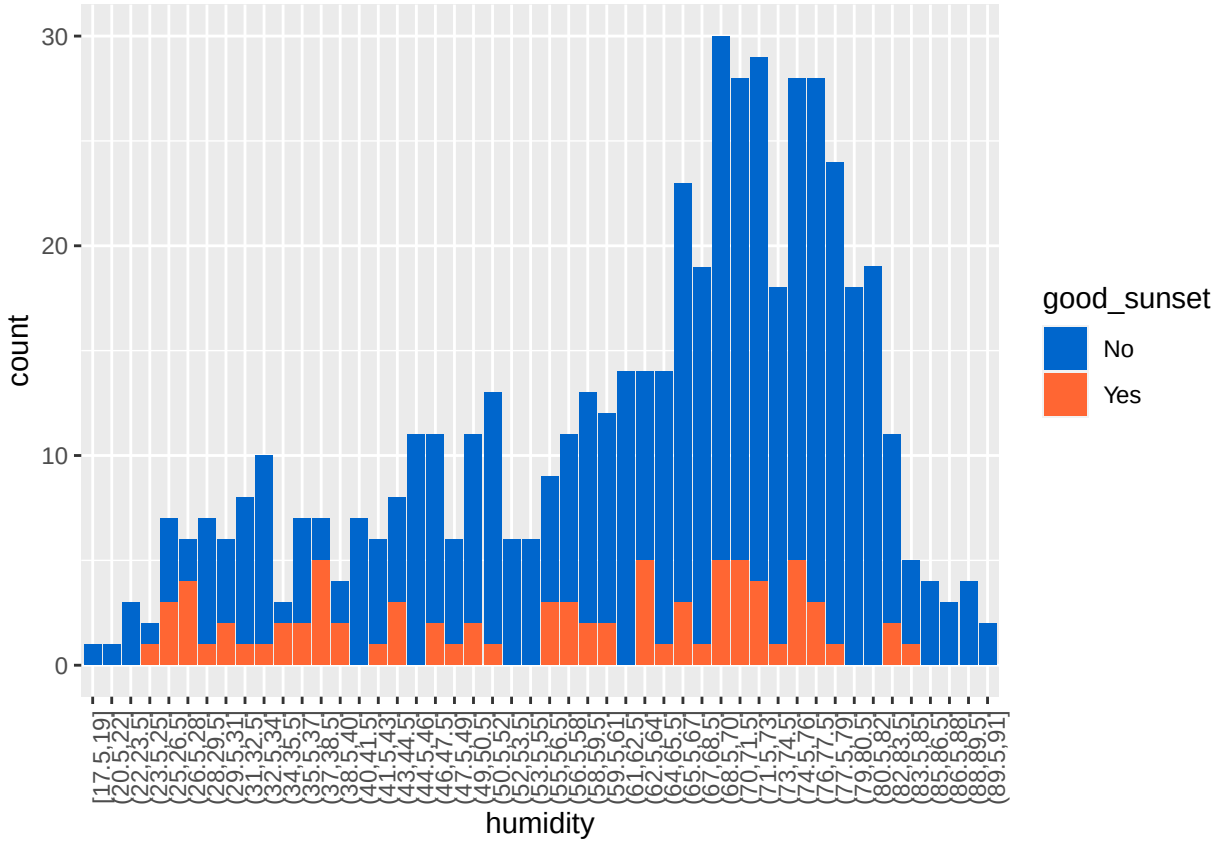
```
  scale_fill_manual(values = c("#0066CC", "#FF6633")) +
  theme(axis.text.x = element_text(angle = 90))
```



Humidity

It is known that humidity will affect the quality of sunsets and their colors. Lower humidity will produce more vibrant colors, whereas high humidity conditions will fade the colors because of the water content in the atmosphere. In the plot below we see that indeed there is a higher concentration of goo sunsets when the humidity is lower, and a much higher concentration of poorer sunsets when the humidity is higher.

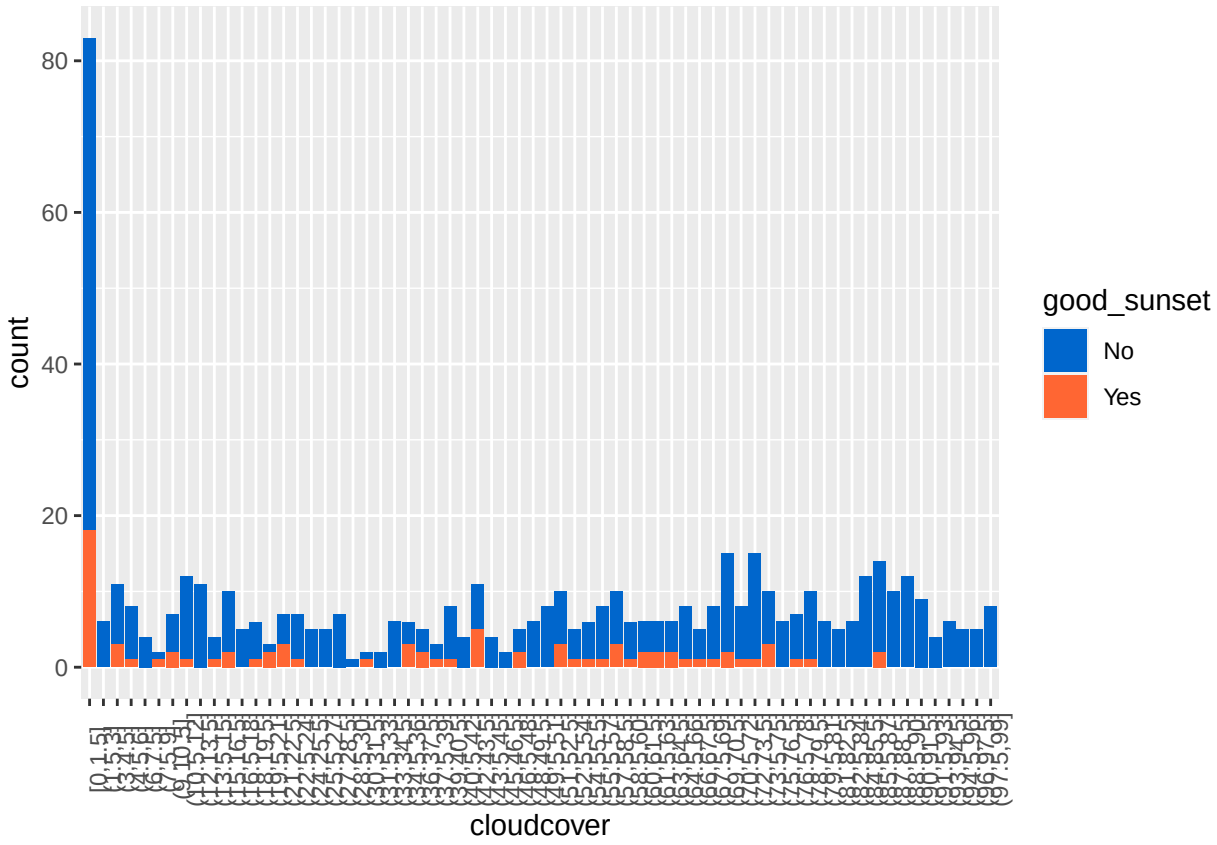
```
sunset_data %>%
  dplyr::select("humidity", "good_sunset") %>%
  dplyr::mutate(humidity = cut(humidity, breaks =
                               seq(min(humidity), max(humidity), by = 1.5),
                             include.lowest = TRUE)) %>%
  group_by(humidity) %>%
  na.omit(humidity) %>%
  ggplot(aes(humidity)) +
  geom_bar(aes(fill = good_sunset)) +
  scale_fill_manual(values = c("#0066CC", "#FF6633")) +
  theme(axis.text.x = element_text(angle = 90))
```



Cloud Cover

One can imagine that if the skies are cloudy, then the sunset will not be great because there won't be much of a sunset in the first place. On the flip side, if there are no clouds at all, the sunset will be pristine, but it may lack the added decoration and added touch of color from the setting sun's reflection on some scattered clouds. Too many clouds is bad, but too few may not be great either, so where's the sweet spot? Well, as we can see below, while a majority of both good and bad sunsets occur when the cloud cover is 0%, a lot of good sunsets seem to occur between a cloud cover of 25% and 50%.

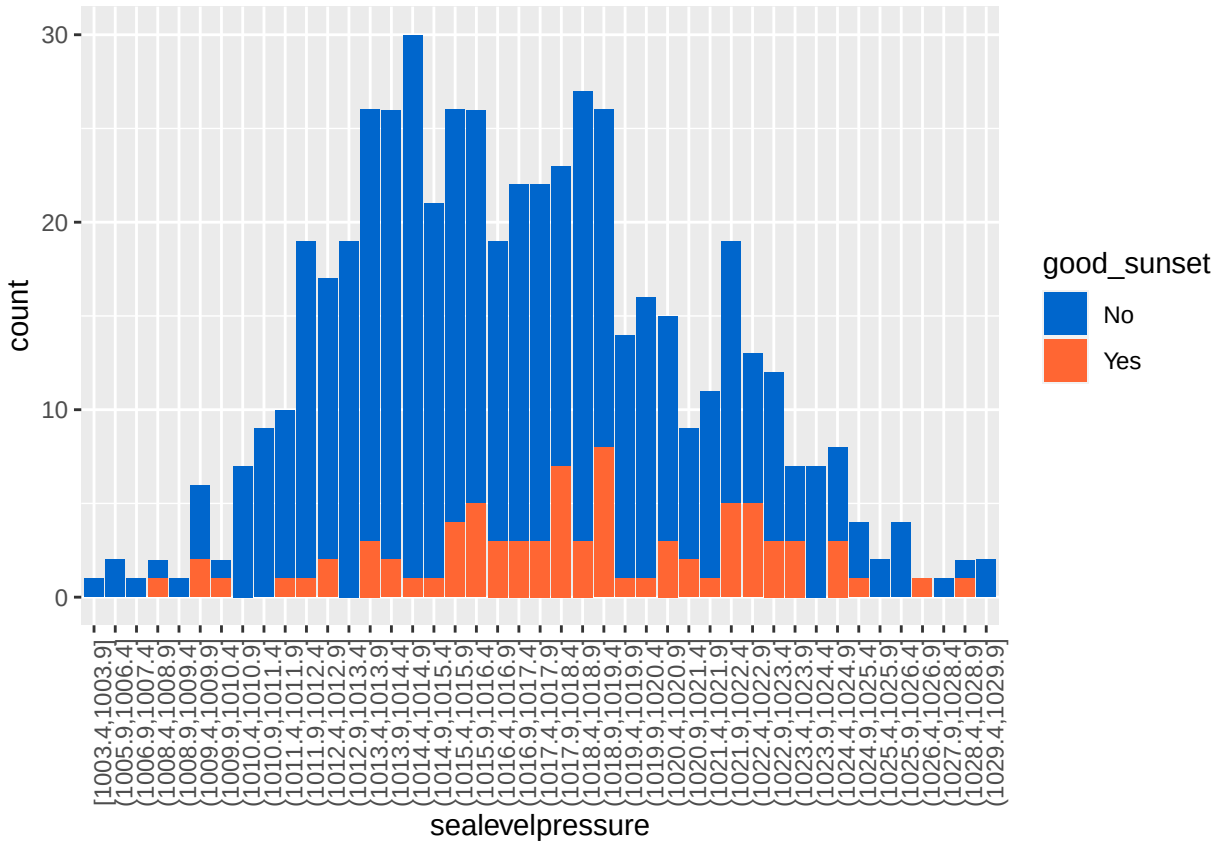
```
sunset_data %>%
  dplyr::select("cloudcover", "good_sunset") %>%
  dplyr::mutate(cloudcover = cut(cloudcover, breaks =
                                   seq(min(cloudcover), max(cloudcover), by = 1.5),
                                 include.lowest = TRUE)) %>%
  group_by(cloudcover) %>%
  na.omit(cloudcover) %>%
  ggplot(aes(cloudcover)) +
  geom_bar(aes(fill = good_sunset)) +
  scale_fill_manual(values = c("#0066CC", "#FF6633")) +
  theme(axis.text.x = element_text(angle = 90))
```



Sea Level Pressure

As for sea level pressure, the quality of a sunset may vary. This is less of a deciding factor of the quality itself, but more of the colors of the sunset. High pressure sunsets are often a brilliant red while low pressure sunsets are a golden-yellow. As can be seen below, good sunsets seems to be normally distributed along sea level pressure but are skewed slightly left in favor of higher sea lever pressure, while poorer sunsets tend to be more present at lower pressures.

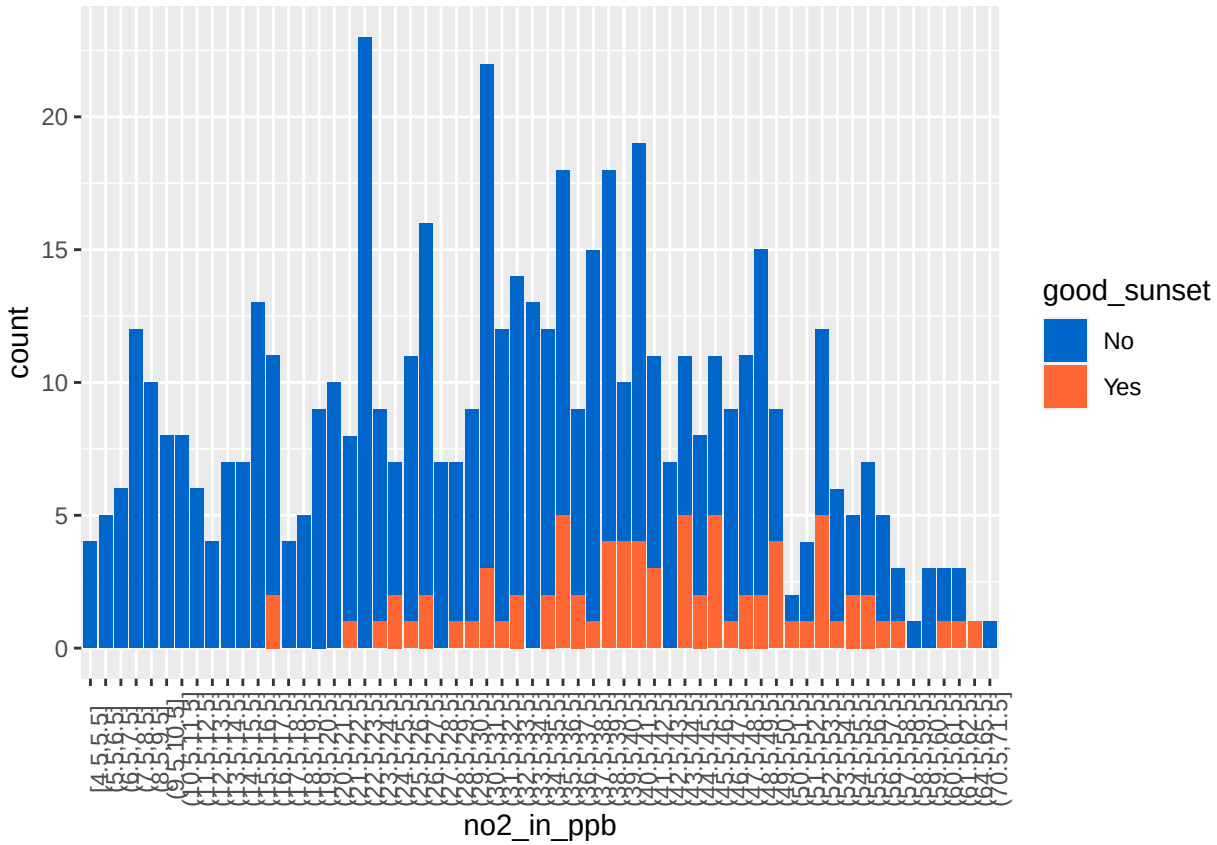
```
sunset_data %>%
  dplyr::select("sealevelpressure", "good_sunset") %>%
  dplyr::mutate(sealevelpressure = cut(sealevelpressure, breaks =
                                seq(min(sealevelpressure), max(sealevelpressure), by = 0.5),
                              include.lowest = TRUE)) %>%
  group_by(sealevelpressure) %>%
  na.omit(sealevelpressure) %>%
  ggplot(aes(sealevelpressure)) +
  geom_bar(aes(fill = good_sunset)) +
  scale_fill_manual(values = c("#0066CC", "#FF6633")) +
  theme(axis.text.x = element_text(angle = 90))
```



Nitrogen Dioxide (NO2)

One that I did not expect myself, was how the nitrogen dioxide levels in the air could affect a sunset's quality. This is because nitrogen and oxygen make up most of the molecules in our atmosphere, so any gas or aerosol in the air will scatter rays of sunlight into separate wavelengths of light. Therefore, when there are more aerosols in the atmosphere, more sunlight is scattered, which results in more colorful skies and more vivid sunsets. As we can see, when the nitrogen dioxide levels in parts per billion in the air are higher, there tend to be more good sunsets, while more poorer sunsets occur when the nitrogen dioxide levels are lower.

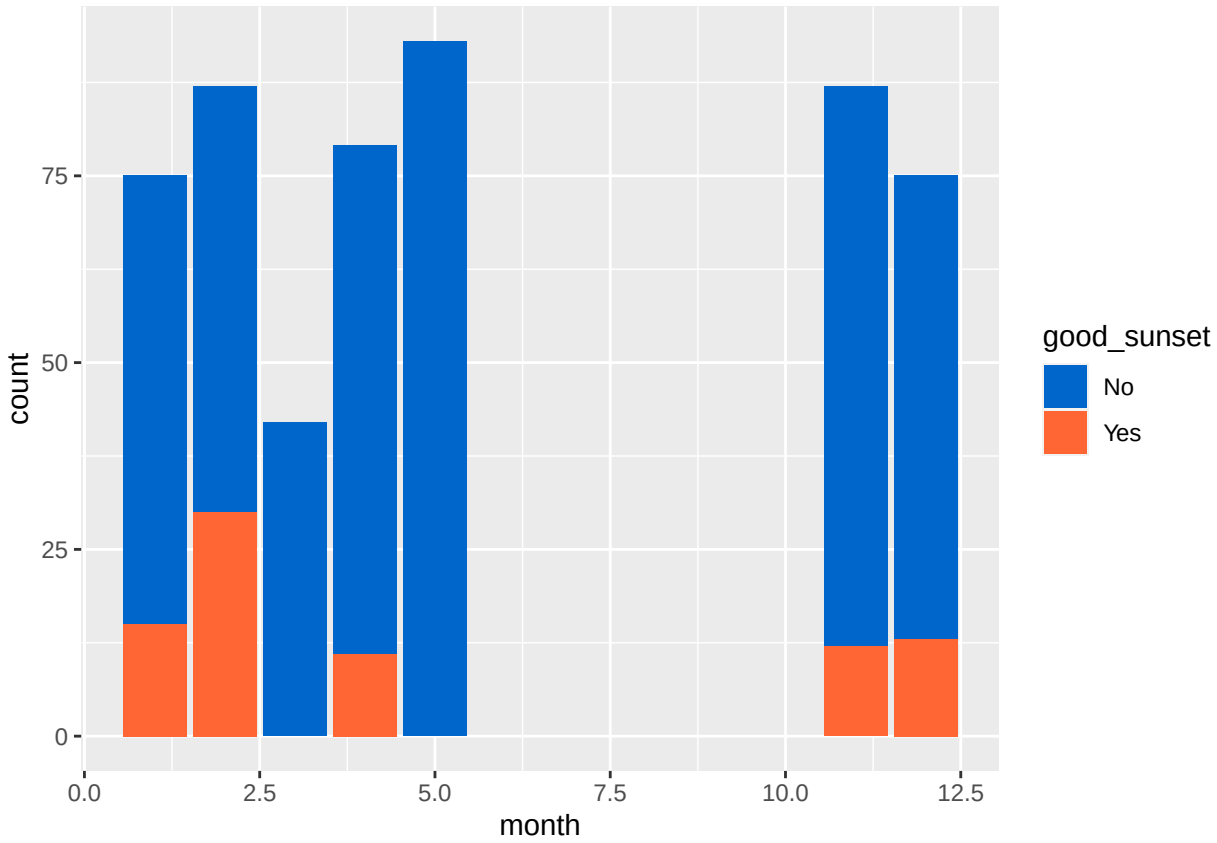
```
sunset_data %>%
  dplyr::select("no2_in_ppb", "good_sunset") %>%
  dplyr::mutate(no2_in_ppb = cut(no2_in_ppb, breaks =
                                 seq(min(no2_in_ppb), max(no2_in_ppb), by = 1),
                               include.lowest = TRUE)) %>%
  group_by(no2_in_ppb) %>%
  na.omit(no2_in_ppb) %>%
  ggplot(aes(no2_in_ppb)) +
  geom_bar(aes(fill = good_sunset)) +
  scale_fill_manual(values = c("#0066CC", "#FF6633")) +
  theme(axis.text.x = element_text(angle = 90))
```



Month

Finally, let's take a look at how the month during which the sunset took place may affect its quality. We previously said that winter sunsets are more vivid, and we can see below that the good sunsets are indeed more present during the months of January, February, November and December, which are the winter months. Since no data was recorded for the months between May and November, these are empty on the plot.

```
ggplot(sunset_data, aes(month)) +
  geom_bar(aes(fill = good_sunset)) +
  scale_fill_manual(values = c("#0066CC", "#FF6633"))
```



Setting Up Models

Now that we have a better idea of how most important variables affect how good a sunset is, we can finally move on to building our models. We will randomly split our data into training and testing sets, set up and create our recipe, and establish cross-validation within our models.

Train/Test Split

As we approach the building of our models, we first need to split our data into separate datasets. One will be used for the training of our models, and one will be the testing set, which is saved till the end and used only once when we actually test our models. Our very first step, is to set our seed so that our random split can be reproduced every time we train our models. Next, we will perform a training / testing split on our data, and stratify on our response variable, `good_sunset`.

```
set.seed(8488)

sunset_split <- initial_split(sunset_data, prop = 0.75,
                              strata = "good_sunset")

sunset_train <- training(sunset_split)
sunset_test <- testing(sunset_split)
```

Dimensions of our sunset training dataset:

```
dim(sunset_train)
```

```
## [1] 402  31
```

Dimensions of our sunset testing dataset:

```
dim(sunset_test)
```

```
## [1] 136  31
```

For splitting the data, I chose a proportion of 0.75 because it allows for more training data, while retaining enough data to be tested since there is a limited amount of observations. The training data has 402 observations while the testing data has 136 observations.

Building Our Recipe

We will now bring together our predictors and our response variable to build our recipe which we will use for all the models. Think of this recipe like an artist's palette. Each variable is a different color that the painter uses to make their painting, in this case, a painting of a sunset. Each color, like each variable, plays an important part in creating the final masterpiece. If you recall Bob Ross' old painting videos, where he would describe each color at the beginning of his painting before mixing them together to make his work of art. Now imagine Bob Ross painting a sunset. We are doing the same here with our predictors; we are putting them together in our recipe to paint our sunset prediction models.

We will be using 20 out of our 24 predictor variables in our recipe. You can think of this like we are using 20 different colors to make our sunset painting. We will be excluding `year` because there is very little difference between the years 2015 and 2016, so this will have no effect on our model. We are also excluding `feelslikemax`, `feelslikemin`, and `feelslike`, because these temperature measurements are already represented by the `tempmax`, `tempmin`, and `temp`, variables in our dataset. Below, you can see our fully completed recipe.

```
# building our recipe
sunset_recipe <- recipe(good_sunset ~ month + co_in_ppm + no2_in_ppb +
                          so2_in_ppb + ozone_in_ppm + aqi + tempmax + tempmin +
                          temp + dew + humidity + precip + precipprob +
                          precipcover + windspeed + winddir + sealevelpressure +
                          cloudcover + visibility + moonphase,
                        data=sunset_train) %>%
  step_scale(all_predictors()) %>% # standardizing our predictors
  step_center(all_predictors())
```

K-Fold Cross Validation

We'll stratify our cross validation on our response variable, `good_sunset`, as well as use 10 folds to perform stratifies cross validation.

```
sunset_folds <- vfold_cv(sunset_train, v = 10, strata = good_sunset)
```

Because building models take so much computing time, we will be saving the results to an RDA file. This is done so that once we have the model we want, we can go back anytime later and load it.

```
save(sunset_folds, sunset_recipe, sunset_train, sunset_test, file = "C:/Users/jules/OneDrive/Desktop/Su
```

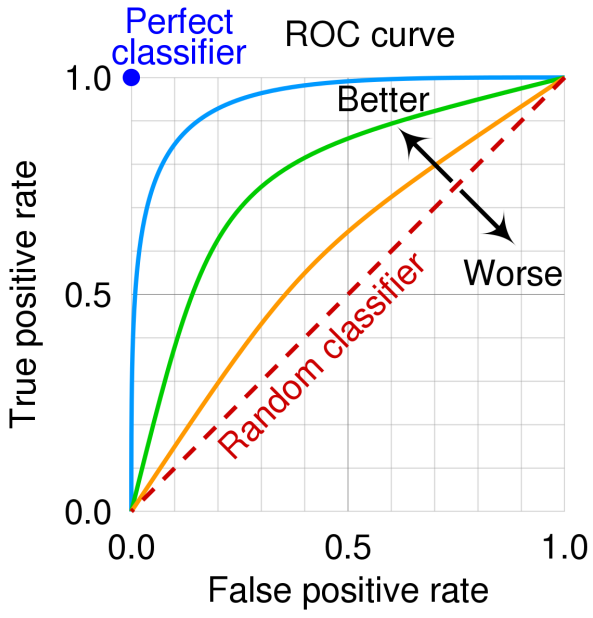
Building Prediction Models

It is now finally time to build our models! Luckily for us, since our dataset is relatively small, none of the models took very long to run, which I was able to use to my advantage in order to run multiple variations of our models, and tune different hyperparamaters with varrying ranges in order to produce the best performing models. However, these models still require quite a bit of computing power, and could therefore not be run directly in this R markdown file. To solve this, each individual model was ran in a separate R file with the loaded data saved above. Each model was then saved in RDA files and will be loaded below for our explorations of the results. Additionally, these models' files and data can be found attached to this project in the *Models* and *RDA* folders in the respective GitHub repository which can be found here: Sunset Prediction Project.

As stated earlier, we fit seven different models. These models were Logistic Regression, Linear Discriminant Analysis, Quadratic Discriminant Analysis, Lasso, Decision Tree, Random Forest, and Support Vector Machine models. The first three models are relatively simple and will take much less time to run. The final four models are the ones we are more interested in, particularly the Random Forest and Support Vector Machines model, as they are better fit for binary classification problems, which is exactly our case.

Performance Metric

To evaluate performance, the two most useful tools in our case are `accuracy` and `roc_auc`. The one we will be utilizing as the metric of performance is `roc_auc`, because it shows the most significant level of efficiency in a binary classification model where the data is not perfectly balanced. To explain further, the `roc_auc` metric, denoted as ROC AUC, is a method of calculating the area under the curve (AUC) for the receiver operating characteristic (ROC) curve, which is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied. Below are examples of ROC curves.



Above is a visualization to help better understand a ROC curve. The area between the curve and the *Random Classifier* line is the AUC score, which increases as the curve approaches a *Perfect Classifier*. While

the curve is a visualization of the ROC AUC score, the numerical AUC value between 0.5 and 1.0 is the determining metric of performance.

Model Building Process

Most models tend to follow a similar process, with the exceptions of the Logistic Regression, Linear Discriminant Analysis, and Quadratic Discriminant Analysis models, which are simpler and quicker. The general workflow of the model building process constituted of the following steps:

1. We first set up the model by specifying what type of model it is, then we set its engine and set its mode. In our case, our mode was always set to 'classification' due to the nature of our project goal.

2. We then set up the workflow for the model, add the new model, and add our established sunset recipe.

We will skip the steps 3-5 for Logistic Regression, Linear Discriminant Analysis, and Quadratic Discriminant Analysis, since they are simpler models that do not require hyperparameters to be tuned.

3. We then set up the tuning grid with the parameters that we want tuned, as well as set the ranges for how many different levels of tuning we want for each parameter.
4. We will then tune the model with the specific hyperparameters of choice.
5. After which we will select the most accurate model from the tuning grid, and then finalize the workflow with those specific tuning parameters.
6. We then fit that model with our workflow to our sunset training dataset.
7. Finally, we will save our results to an RDA file in order to load them back into our main project file.

Model Results

Since our dataset was on the smaller side with only 538 observations, our models took less time to run then they would with much larger datasets. We now have all of our models completed and saved as well as their individual outcomes and scores. We will now load in the saved results of each model and start analyzing their individual performances.

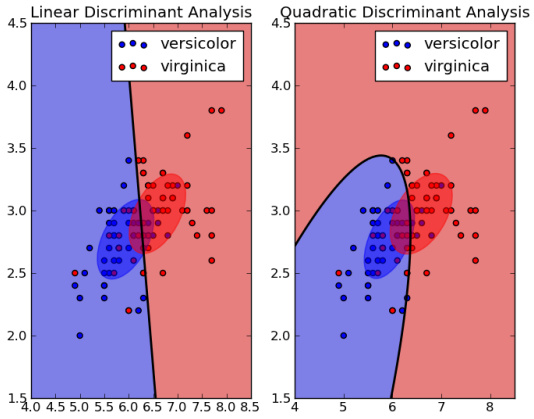
```
load("C:/Users/jules/OneDrive/Desktop/Sunset-Prediction-Project/RDA/sunset_Model_Setup.rda")
load("C:/Users/jules/OneDrive/Desktop/Sunset-Prediction-Project/RDA/sunset_Logistic_Regression.rda")
load("C:/Users/jules/OneDrive/Desktop/Sunset-Prediction-Project/RDA/sunset_Linear_Discriminant.rda")
load("C:/Users/jules/OneDrive/Desktop/Sunset-Prediction-Project/RDA/sunset_Quadratic_Discriminant.rda")
load("C:/Users/jules/OneDrive/Desktop/Sunset-Prediction-Project/RDA/sunset_Lasso_Regression.rda")
load("C:/Users/jules/OneDrive/Desktop/Sunset-Prediction-Project/RDA/sunset_Decision_Tree.rda")
load("C:/Users/jules/OneDrive/Desktop/Sunset-Prediction-Project/RDA/sunset_Random_Forest.rda")
load("C:/Users/jules/OneDrive/Desktop/Sunset-Prediction-Project/RDA/sunset_Support_Vector_Machine.rda")
```

Visualizing Results

One of the most useful tools for visualizing the results of models that have been tuned is the `autoplot` function in r. This will visualize the effects that the change in certain parameters has on our metric of choice, `roc_auc`. For the sake of space, we only display the plots of our models with the three best ROC AUC values, which will be explored and displayed in the next section.

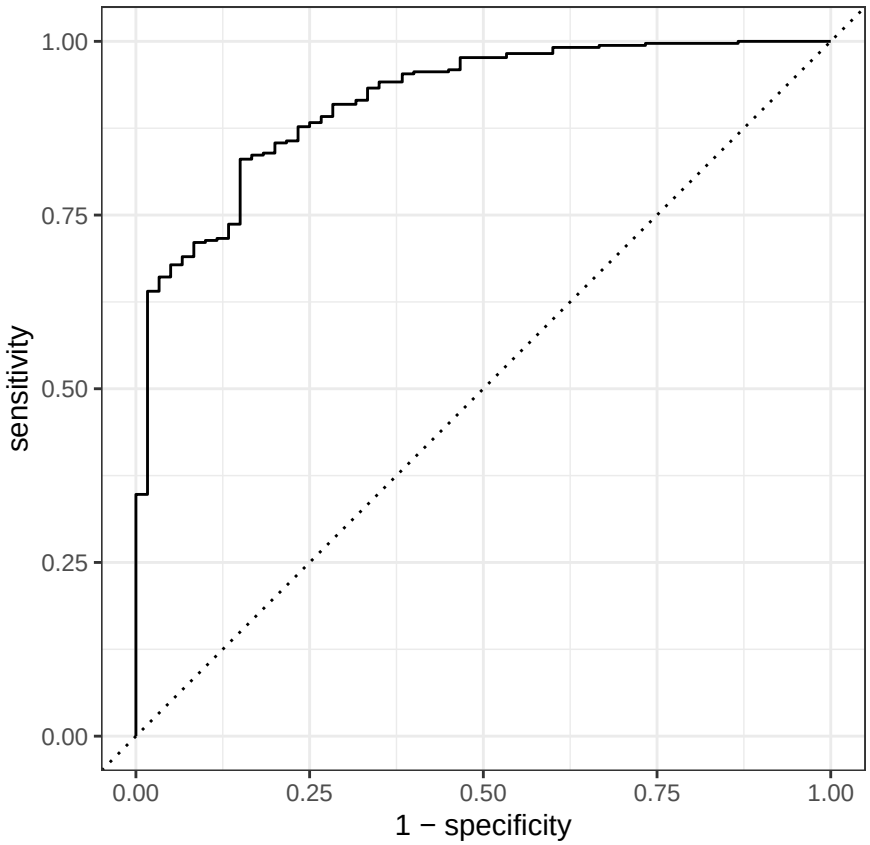
Quadratic Discriminant Analysis

A quadratic discriminant analysis is a statistical classifier model that uses a quadratic decision surface to separate measurements of two or more classes of objects or events. It is a more advanced version of a linear model as it is used to find a non-linear boundary between our classifiers, and assumes that each class follows a Gaussian distribution. Below, we can see a visualization of a linear discriminant analysis (LDA) model and a quadratic discriminant analysis (QDA) model. While the LDA model seeks a linear boundary between the classifier data, the QDA model displays a more effective and accurate curved boundary.



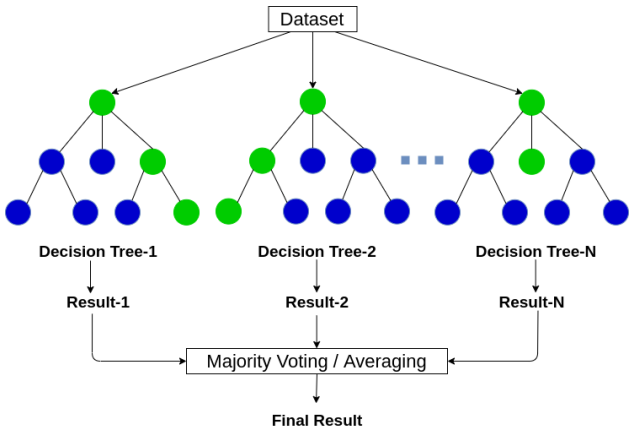
Surprisingly enough, as we can see in the ROC curve plot below, our Quadratic Discriminant Analysis Model did a relatively good job of predicting our sunsets when it was fit back on our training data.

```
sunset_roc_qda %>%
  roc_curve(good_sunset, .pred_No) %>%
  autoplot()
```



Random Forest

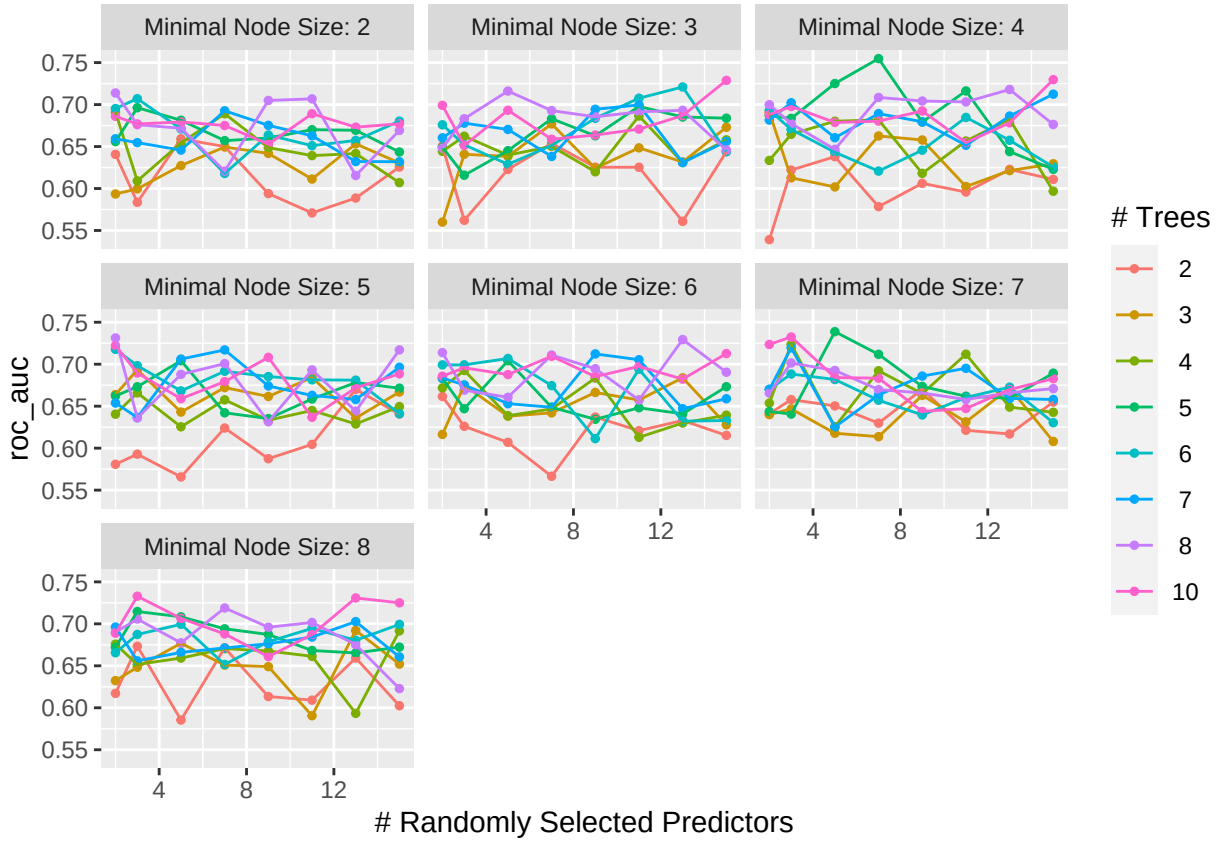
A random forest model is a supervised ensemble learning technique that consists of numerous decision trees. Decision tree models tend to struggle with over fitting the training data, which is overcome in a random forest by averaging the prediction results of each decision tree and determining a final output. The random forest algorithm stacks multiple classifiers together to improve its overall performance. To get a better idea of how a random forest model works, I have provided a visualization of its process below.



In our random forest model, we are tuning three different hyperparameters. These parameters are:
- `mtry` which represents the amount of predictors that will be sampled randomly during the creation of the

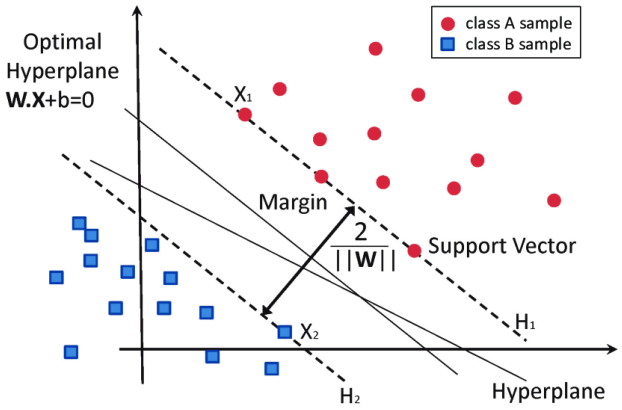
models,
- `trees` which represents the amount of trees present in the random forest model,
- `min_n` which represents the minimum amount of data values required to be in a tree node in order for it to be split further down the tree.
The ROC AUC scores seem to vary quite a lot depending on the number of trees, but overall it is clear that more trees leads to a higher ROC AUC score. The optimal node size was 4, present in the middle plot, with 5 trees, and 7 randomly selected predictors. Unless our next model can surpass this, the random forest seems to be the best model.

```
autoplot(sunset_rf_tune_res_auc)
```



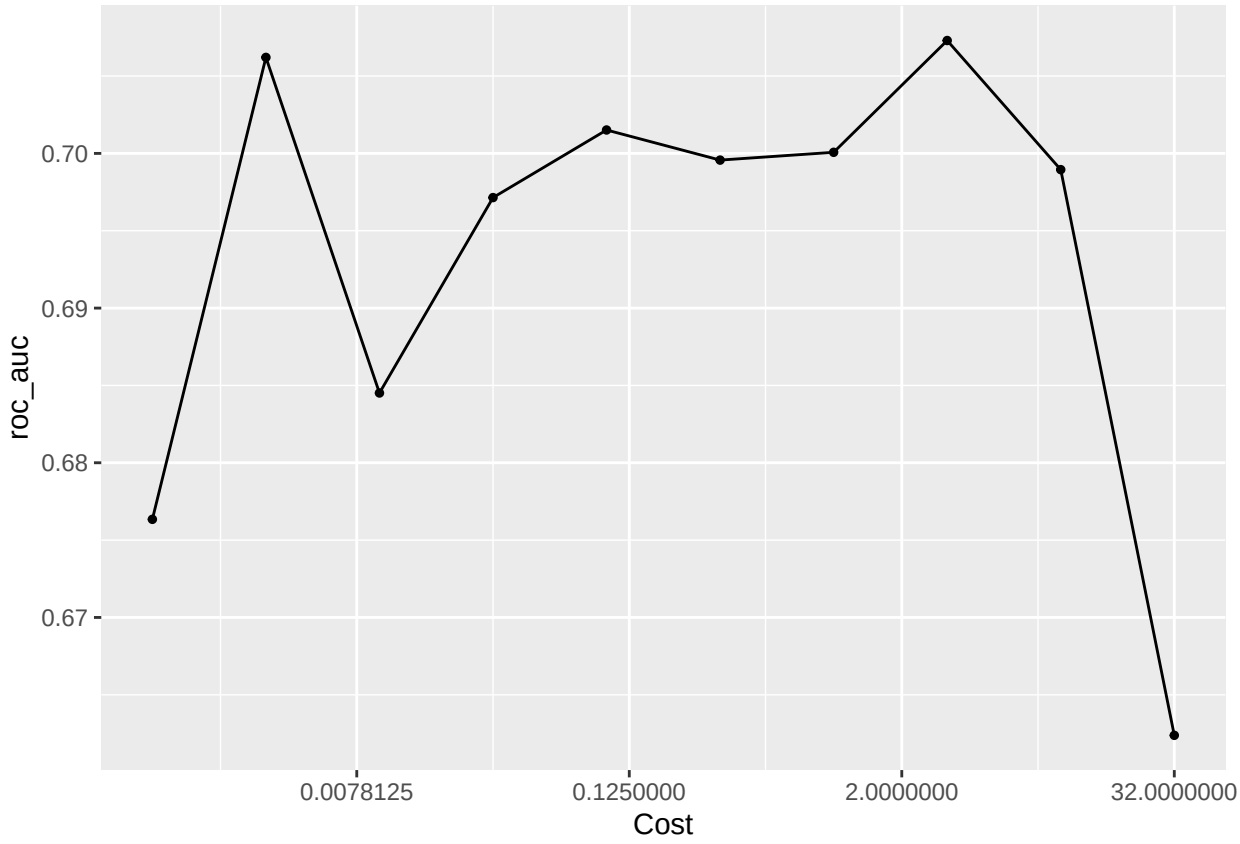
Support Vector Machine (SVM)

Our very last model is a support vector machine. A support vector machine algorithm is mainly used for binary classification tasks, like ours, and produces significant accuracy with less computation power. An SVM model plots each data observation as a point in n-dimensional space, where n is a number of features we have, with the value of each feature being the value of a particular coordinate. The support vectors are simply the coordinates of each individual observation. Classification is then performed by finding the hyperplane that differentiates the two classes best. Below is a diagram to visualize how a support vector machine proceeds and what the hyperplanes look like.



Like most of our models, the actual process of the algorithm is much more complicated, but we can see in our plot below that while our SVM model did relatively well, in our case it did not attain a higher ROC AUC score than our random forest model. However since support vector machine algorithms are still very powerful, it will still be worth exploring its testing results later on.

```
autoplot(sunset_svm_auc_tune_res)
```



Model Accuracies

In order to summarize the best ROC AUC values from our seven models, we will create a tibble (think like a table, but better) in order to display the estimated final `roc_auc` value for each fitted model.

```r
sunset_log_reg_auc <- augment(sunset_log_fit, new_data = sunset_train) %>%
  roc_auc(good_sunset, estimate = .pred_No) %>%
  select(.estimate)

sunset_lda_auc <- augment(sunset_lda_fit, new_data = sunset_train) %>%
  roc_auc(good_sunset, estimate = .pred_No) %>%
  select(.estimate)

sunset_qda_auc <- augment(sunset_qda_fit, new_data = sunset_train) %>%
  roc_auc(good_sunset, estimate = .pred_No) %>%
  select(.estimate)

sunset_lasso_auc <- augment(sunset_lasso_final_fit, new_data = sunset_train) %>%
  roc_auc(good_sunset, estimate = .pred_No) %>%
  select(.estimate)

sunset_decision_tree_auc <- augment(sunset_dt_final_fit, new_data = sunset_train) %>%
  roc_auc(good_sunset, estimate = .pred_No) %>%
  select(.estimate)

sunset_random_forest_auc <- augment(sunset_rf_final_fit_auc, new_data = sunset_train) %>%
  roc_auc(good_sunset, estimate = .pred_No) %>%
  select(.estimate)

sunset_svm_auc <- augment(sunset_svm_final_fit_auc, new_data = sunset_train) %>%
  roc_auc(good_sunset, estimate = .pred_No) %>%
  select(.estimate)


sunset_roc_aucs <- c(sunset_log_reg_auc$.estimate,
                           sunset_lda_auc$.estimate,
                           sunset_qda_auc$.estimate,
                           sunset_lasso_auc$.estimate,
                           sunset_decision_tree_auc$.estimate,
                           sunset_random_forest_auc$.estimate,
                           sunset_svm_auc$.estimate)

sunset_mod_names <- c("Logistic Regression",
            "LDA",
            "QDA",
            "Lasso",
            "Decision Tree",
            "Random Forest",
            "Support Vector Machine")
```

```r
sunset_results <- tibble(Model = sunset_mod_names,
                             ROC_AUC = sunset_roc_aucs)

sunset_results <- sunset_results %>%
  dplyr::arrange(-sunset_roc_aucs)

sunset_results
```

```
## # A tibble: 7 x 2
##   Model                  ROC_AUC
##   <chr>                    <dbl>
## 1 Random Forest            0.990
## 2 Support Vector Machine   0.961
## 3 QDA                      0.914
## 4 Logistic Regression      0.835
## 5 LDA                      0.835
## 6 Lasso                    0.824
## 7 Decision Tree            0.817
```

As we can see in our tibble, the Random Forest model performed the best overall with a ROC AUC score of 0.9901, with the support vector machine close behind at 0.9612. Of course, this is only fitted on the training data, so our models still need to perform on our testing data that we have reserved for exactly this. We will be moving forward with this Random Forest model, but we will also explore the support vector machine model's performance on the testing data for the sake of exploration. Let's now find their true performances on our testing dataset.

Results From Our Best Models

Now that we know that our best model was a random forest, we can progress to analyzing its true results. As previously stated, we will also be analyzing the results of our support vector machines model. While it did not have best ROC AUC performance on the training data compared to the random forest, the support vector machine algorithm is still incredibly powerful and worth exploring further to discover its scores on the testing data.

Random Forest Model

Since our random forest model had the best overall performance, we now want to examine how strong it is on data it has not seen yet. The high ROC AUC scores you saw above were the models' ability to predict a sunset's quality using the same data it was originally trained on, thus explaining its strong results.

And the Best Model is ...

Random Forest 156! The random forest model #156 seemed to have performed the best overall from all the random forest models. This is on top of being the best of the seven different prediction models. Below is the model's output and scores, as well as the its associated parameters.

```
show_best(sunset_rf_tune_res_auc, metric = "roc_auc") %>%
  select(-.estimator, .config) %>%
  slice(1)
```

```
## # A tibble: 1 x 8
##    mtry trees min_n .metric  mean     n std_err .config
##   <int> <int> <int> <chr>   <dbl> <int>   <dbl> <chr>
## 1     7     5     4 roc_auc 0.755    10  0.0360 Preprocessor1_Model156
```

Now that we have our best overall model, we can finally fit it to our testing data and discover its actual performance in predicting the aesthetic quality of our sunsets.

Final ROC AUC Results

Now, the moment we have all been waiting for, finding our model #156's true ROC AUC performance results on our testing dataset that we have been saving just for this. Let's take a look!

```
sunset_rf_roc_auc <- augment(sunset_rf_final_fit_auc, new_data = sunset_test, type = 'prob') %>%
  roc_auc(good_sunset, .pred_No) %>%
  select(.estimate)

sunset_rf_roc_auc
```

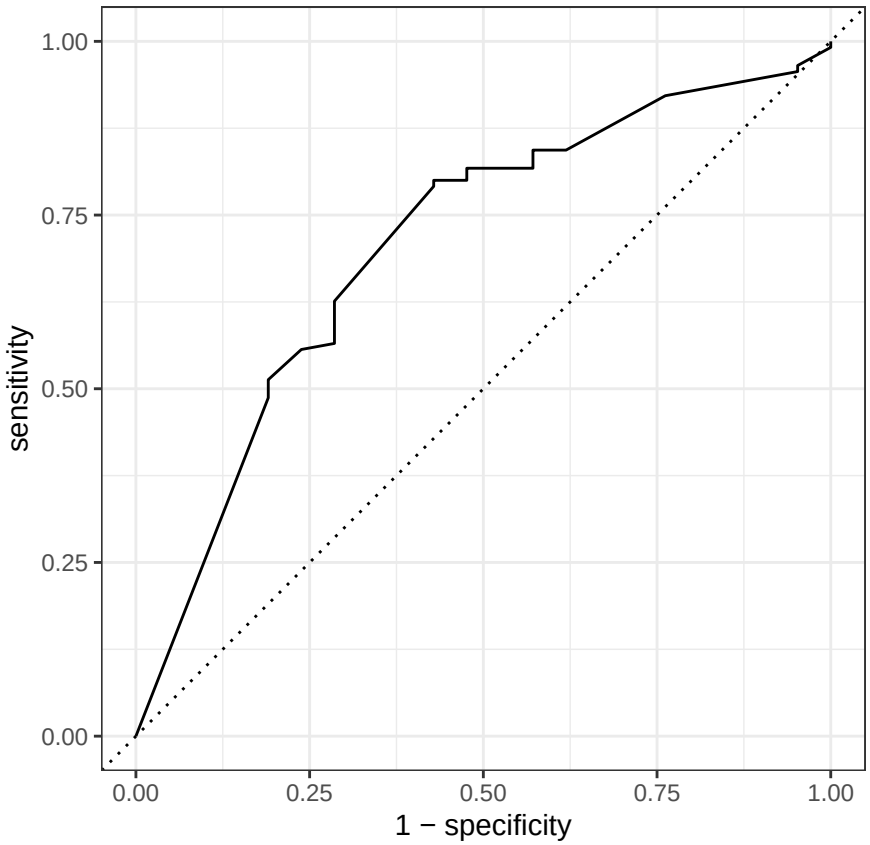
```
## # A tibble: 1 x 1
##   .estimate
##       <dbl>
## 1     0.707
```

With a final ROC AUC score of 0.7075 on our testing data, we can say our model did pretty good! A AUC value between 0.7 and 0.8 is generally considered a fair and acceptable result. This means that the model performed alright, but could have also been much better. In our case, since we are attempting the seemingly ridiculous task of predicting how beautiful a sunset is, I would say this was a success! Weather is already difficult enough to predict, so to be able to say that we can relatively accurately predict the aesthetic quality of a sunset is a huge achievement.

ROC Curve

To visualize our AUC score, we will plot our ROC curve. The higher up and left the curve is, the better the model's AUC will be. As we can see below, while our curve does not perfectly resemble the top left right angle of a square, it still is curving in the right place and therefore confirms our computed AUC score above.

```
augment(sunset_rf_final_fit_auc, new_data = sunset_test, type = 'prob') %>%
  roc_curve(good_sunset, .pred_No) %>%
  autoplot()
```



Support Vector Machine Model

As previously decided, we are also going to explore the results of our support vector machine model on our testing data. We will analyze its true performance by once again computing the ROC AUC score and plotting the ROC curve alongside.

Surprise, Surprise!

Well, to my surprise, the final ROC AUC score of our support vector machine algorithm was higher than our random forest model!

The computed Support Vector Machine model ROC AUC score on the testing dataset is as follows. As we can notice, it is only about 0.008 higher than the random forest model, which is not a huge margin of difference, but can still very statistically significant in the world of statistics and ROC AUC scores.

```
sunset_svm_roc_auc <- augment(sunset_svm_final_fit_auc, new_data = sunset_test, type = 'prob') %>%
  roc_auc(good_sunset, .pred_No) %>%
  select(.estimate)

sunset_svm_roc_auc
```

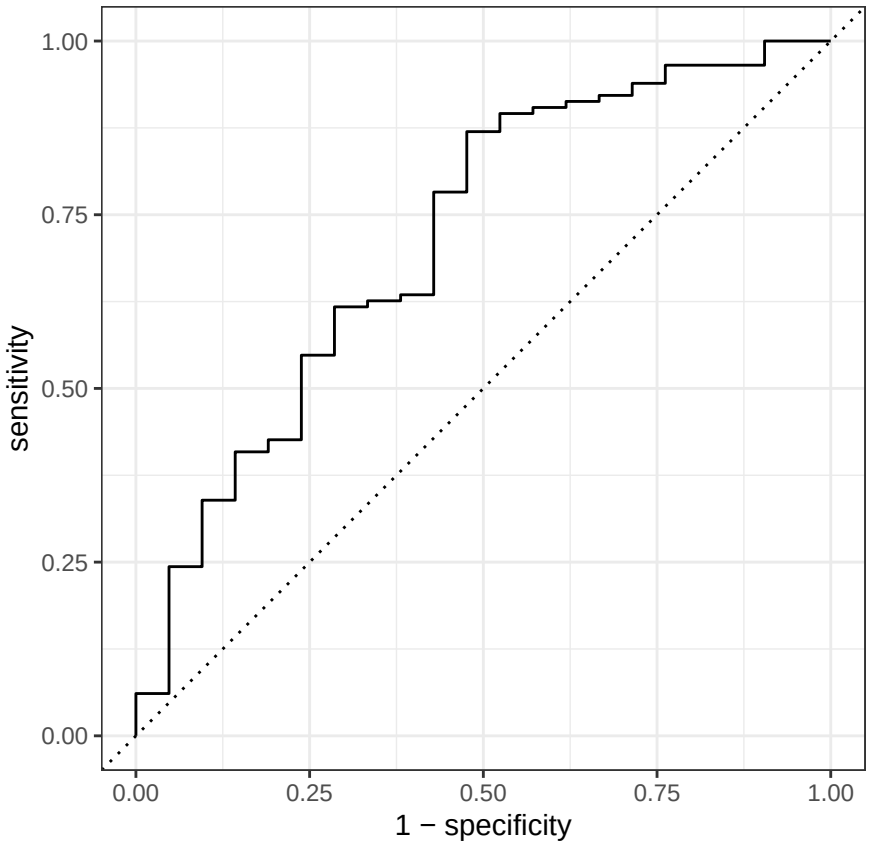
```
## # A tibble: 1 x 1
##   .estimate
##       <dbl>
## 1     0.716
```

With a final ROC AUC score of 0.7155, we can officially say that our support vector machine model performed better than the random forest model on our testing data.

To visualize this result, we will once again plot the ROC curve. While our curve does not exactly look like the Greek letter Γ like we would want, it is still curving up and towards the left, thus providing a good visualization to accompany our ROC AUC score. This proves that our support vector machine did indeed perform better on the testing data than any other model.

```
sunset_svm_roc_curve <- augment(sunset_svm_final_fit_auc, new_data = sunset_test, type = 'prob') %>%
  roc_curve(good_sunset, .pred_No) %>%
  autoplot()

sunset_svm_roc_curve
```



Why is this Happening?

There is a multitude of reasons why we may be seeing these results. The reason is most likely a combination of multiple. Support Vector Machines are known to be very powerful classification models, and work better than most prediction algorithms when provided with limited data. In our case, since we only have 538 observations, it would make sense that a support vector machine would produce a better output than a random forest which thrives on larger datasets.

Additionally, support vector machines perform well on all sorts of data, while random forests are generally better fit for data with both numerical and categorical data. Our particular sunset dataset that we used in our recipe and our models includes only numerical data, which could be negatively impacting the performance of the random forest, thus making our support vector machines the better choice here.

Overall, it seems that the difference in performance between a support vector machine and a random forest lies in its application, the type of data its provided with, and the end goal. In our case of creating a binary classification model with limited data, the support vector machine is the more performant model.

Putting the Model to the Test

Now that we have our model completed and polish, let's put it to use. How useful is it in predicting good sunsets? We will start by providing it with sample data and seeing whether or not it predicts a good sunset.

Beautiful Sunset Test

Remember that stunning sunset picture at the top of the document? Well, I collected all the necessary meteorological and air quality data from that particular day in Santa Barbara, so let's see if our model predicts that it is indeed a beautiful sunset.

```
sunset_good_test_example <- data.frame(
  month = 12,
  co_in_ppm = 1.8,
  no2_in_ppb = 62.0,
  so2_in_ppb = 1.5,
  ozone_in_ppm = 0.015,
  aqi = 60,
  tempmax = 82.3,
  tempmin = 54.0,
  temp = 66.3,
  dew = 37.9,
  humidity = 38.0,
  precip = 0.00,
  precipprob = 0,
  precipcover = 0.00,
  windspeed = 3.8,
  winddir = 276.3,
  sealevelpressure = 1016.6,
  cloudcover = 0.4,
  visibility = 9.6,
  moonphase = 0.96
)
```

```
predict(sunset_rf_final_fit_auc, sunset_good_test_example, type = "class")
```

```
## # A tibble: 1 x 1
##   .pred_class
##   <fct>
## 1 Yes
```

Woohoo! As we can see, the model correctly predicted that the sunset would be beautiful. This is very encouraging, because it means that our model is successful. While the ROC AUC score already demonstrated that our model performed well, it is reassuring to watch it succeed when put to the test.

Bad Sunset Example

For the sake of confirming our results, let's test our model on data that would be associated with a poorer sunset, and make sure that our model correctly predicts the result.

```
sunset_bad_test_example <- data.frame(
  month = 5,
  co_in_ppm = 1.2,
  no2_in_ppb = 41.3,
  so2_in_ppb = 1.0,
  ozone_in_ppm = 0.062,
  aqi = 39,
  tempmax = 83.9,
  tempmin = 59.7,
  temp = 69.9,
  dew = 51.1,
  humidity = 55.0,
  precip = 0.00,
  precipprob = 0,
  precipcover = 0.00,
  windspeed = 7.4,
  winddir = 222.3,
  sealevelpressure = 1015.3,
  cloudcover = 0.3,
  visibility = 9.7,
  moonphase = 0.65
)
```

```
predict(sunset_rf_final_fit_auc, sunset_bad_test_example, type = "class")
```

```
## # A tibble: 1 x 1
##   .pred_class
##   <fct>
## 1 No
```

Success again! Now we know for sure that our model is able to predict both good and bad sunsets. This is to be taken with a grain of salt of course, since our model accuracy is obviously not perfect. This means that it is possible that the model could be fed data associated with a beautiful sunset, and it could output a result labeling it as a bad sunset. However, I feel confident saying that our model was not only successful, but was also a great demonstration of the power of machine learning techniques in our current world.

Extension: Final Model Accuracy

While the ROC AUC score was our primary performance metric, I also calculated the `Accuracy` of model #156 on our testing data in order to provide a simpler explanation of our model results.

The Random Forest accuracy:

```
augment(sunset_rf_final_fit_accuracy, new_data = sunset_test, type = 'prob') %>%
  accuracy(good_sunset, .pred_class) %>%
  select(.estimate)
```

```
## # A tibble: 1 x 1
##   .estimate
##       <dbl>
## 1     0.831
```

Our random forest model was able to predict beautiful sunsets in our testing data with about 83.1% accuracy. That's awesome! I can now confidently tell my friends whether or not a sunset is really worth dropping everything to watch, with about 81.6% accuracy.

The Support Vector Machine accuracy:

```
augment(sunset_svm_final_fit_accuracy, new_data = sunset_test, type = 'prob') %>%
  accuracy(good_sunset, .pred_class) %>%
  select(.estimate)
```

```
## # A tibble: 1 x 1
##   .estimate
##       <dbl>
## 1     0.846
```

With an accuracy of 84.6%, we can see that our support vector machine indeed performed better than our random forest model on our testing data. However, the margin of difference is not huge, which goes to show that both are still powerful and useful models that could still be used to predict the quality of sunsets, just with a random forest model having slightly lower overall accuracy.

Conclusion

Throughout this project, we have researched, explored, and analyzed our data and its variables in order to build and test a model that could predict how beautiful a sunset will be. After diligent analysis, testing, and computing, we can say that the Random Forest model was the best at predicting the aesthetic quality of a sunset. Unfortunately, this model was not perfect, and leaves room for improvement.

As for further extensions, one could consider the application of a neural network. This would most likely require more computation power, depending on the network, and could be applied to more predictors. On top of this, a neural network could be applied directly to the collected Instagram images, and be used to identify which colors are present during a beautiful sunset, such as purple, red, orange, pink, and yellow. This network could then be trained on thousands of these images in order for it to be able to recognize these same colors in pictures it has not seen yet. This could be combined with a model such as this project, in order to achieve an even higher ROC AUC score and accuracy. This could then become automated if paired with a program that self-collects the necessary meteorological and air quality data each day and feeds it to our model in order to develop a sunset prediction for any given day. This is a next step that I will definitely consider, with my current goal being creating it in Python, and perhaps even turning it into a useful tool for the public to use.

Overall, my biggest takeaways from this project lie in my renewed appreciation for sunsets in general. Living in Santa Barbara, I am used to watching beautiful sunsets on a daily basis, but I now understand all the elements that go into painting such a stunning picture every evening. I can now say that I am able to let my friends know when they should drop everything and run to the beach to view another spectacle of nature, for which I am very grateful.

When put into perspective, our model performed much better than expected. Weather is completely random, and therefore already extremely hard to predict. Even with our modern day cutting-edge technology, we still struggle to predict the precipitation for the following day. One can imagine that predicting a sunset would

be even harder, if not impossible. To be able to say that I can somewhat accurately predict how beautiful a sunset will be if I am provided the proper data, is a huge achievement, and is something I am proud of. This was a great opportunity to build my experience and skills with machine learning techniques, as well as explore a subject that is a of great interest and meaning to me.

To top it all off, here is the picture of another beautiful sunset taken from my deck in sunny Santa Barbara: