0.0.1 Question 0: Understanding the Data

Notice that for the table `annual_county_aqi`, the 90th percentile AQI is reported as a column. Why would the 90th percentile AQI be useful as opposed to the maximum? What does it mean when the difference between the 90th percentile AQI and Max AQI is very large compared to the difference between the 90th percentile AQI and the median AQI?

The 90th percentile represents 90% of the values being below the percentile value and 10% of the values being above the percentile value. This makes the 90th percentile more beneficial than the maximum because the maximum could potentially be an outlier, which would not accurately represent the true 'upper end' of the data that is more plausible to occur. The max AQI may be affected by an extreme weather event or other similar factors causing it to skew even though it is not truly representative of expected high range values. When the difference between the 90th percentile AQI and the max AQI is very large compared to the difference between the 90th percentile and the median AQI this means the data could potentially be skewed right and contain extreme outliers. The median is not skewed by outliers like max would be so if the difference between median and 90th percentile is less large compared to max and 90th percentile it further goes to show that we have extreme outliers in our data that are skewing our results.

0.0.2 Question 3b: Worst AQI States

What are the states that are in both of the top 10 lists? Why do you think most of these states are on both of the lists?

The states that were in both lists were California, Arizona, Connecticut, Nevada. Some of these states tend to have the largest population in the United States with very dense cities, which could lead to more traffic and an increase in max AQI and median AQI. These four states are also at elevated risk of wildfires. They tend to have greater fire danger, and usually have higher temperatures as well. Fires release a great deal of ash and pollutants into the air leading to both the worst max and median AQI's. It may also be the case that these states have more refineries or pollutant chemical waste causing an increase of chemical concentrations in the air leading to the rise of air quality index.

Question 4bii: Missing Days Are there any key missing dates in common between the counties that have missing AQI data? What two counties have the most missing days and why do you think they do?

From running this short script to count the number of common dates, we can see that the most common date is '2020-02-29' as it occurred with three counties: Lake, Trinity, and Plumas. This could have occurred because February 29th is a day of the year that only occurs every 4 years, so they could have forgotten that it existed and not taken data for it. The second most common date is '2020-12-02' as it occurred with two counties: Del Norte and Trinity. The two counties with the most missing days are Trinity and Del Norte County. Both counties are located in the northwestern region of California. An interesting period of time with the most consecutive missing dates from both of these counties were from '2020-07-01' to '2020-07-14'. Besides this large time period of missing dates, in general throughout the year these two counties have numerous missing dates. It could possibly be because these counties have very small populations compared to other counties in the datasets and have lower activity that can contribute to a large variation in AQI, and therefore many missing dates of data.

0.0.3 Question 5b: AQI over Time Analysis

Is there anything interesting that you notice in `aqi_per_month_CA`? If so, why do you think that is?

California has a higher average AQI for every month in the year compared to the average of the country per month. California also has a larger total population compared to other states so this higher average AQI result could be due to more vehicular emissions, demand for energy production, household energy consumption, and frequent wildfires across the state. This is also thoroughly explained by our answer to 3b as California is one of the states to have the highest max and median AQI's. There was also an extremely high AQI in September most probably caused by the El Dorado Fire, San Antonio Fire and the extreme heat waves and Santa Ana winds at the time.

0.0.4 Question 5c: Modeling AQI over Time

Based on the AQI pattern in the year 2020, if we were to model AQI over the last 10 years, with the average AQI per year being the same, what sort of parametric function $f(x)$ would you use? Let us say that we see a linear increase in the average AQI per year over the last 10 years instead, then what parametric function $g(x)$ would you use?

Based on the AQI pattern in the year 2020, if we were to model AQI over the last 10 years, with the average AQI per year being the same, what sort of parametric function f(x)f(x) would you use? Let us say that we see a linear increase in the average AQI per year over the last 10 years instead, then what parametric function g(x)g(x) would you use?

aqi_per_month_CA.plot()

aqi_per_month.plot()

From plotting the data and seeing the difference, we can see that over one year the function looks quadratic. But if we assume the pattern stays consistent, then after 10 years the graph would look like a periodic wave graph (such as sine or cosine). If the average AQI increases linearly every year, we can then use a linear graph and linear parametric function.

0.0.5 Question 6c: Analyze Heatmap

Look up where the dark regions correspond to. Does this heatmap make sense?

This heatmap does make sense. Looking at the darker regions, they seem to correspond to areas where fires occurred during September 2020 causing the high median AQI values for that month. For example Yosemite Valley - Yosemite National Park latitude is approximately 37.7 and Longitude approximately -119 which corresponds to a very dark area of the heat map which is very accurate as there were fires in that area during the month of September probably leading to the increase in AQI. Around longitude -117.417931 and latitude 34, we are near Fresno county which had the infamous Creek Fire that lasted for a couple of months in late 2020. The heatmap may be representing counties in California in the month of September impacted by wildfires leading to higher AQI values.

0.0.6 Question 7a - Code and Analysis

Please complete all of your analysis in the **single cell** below based on the prompt above.

```
In [29]: #epa_data_CA.get('daily_ozone')[['Date Local']]
         ozone = epa_data_CA.get('daily_ozone')
         ozone['month'] = pd.to_datetime(epa_data_CA.get('daily_ozone')['Date Local'].values).month

         so2 = epa_data_CA.get('daily_so2')
         so2['month'] = pd.to_datetime(epa_data_CA.get('daily_so2')['Date Local'].values).month

         co = epa_data_CA.get('daily_co')
         co['month'] = pd.to_datetime(epa_data_CA.get('daily_co')['Date Local'].values).month

         no2 = epa_data_CA.get('daily_no2')
         no2['month'] = pd.to_datetime(epa_data_CA.get('daily_no2')['Date Local'].values).month

         ozone = epa_data_CA.get('daily_ozone').groupby('month').agg(np.mean)[['Arithmetic Mean']].renam
         so2 = epa_data_CA.get('daily_so2').groupby('month').agg(np.mean)[['Arithmetic Mean']].rename(co
         co = epa_data_CA.get('daily_co').groupby('month').agg(np.mean)[['Arithmetic Mean']].rename(colu
         no2 = epa_data_CA.get('daily_no2').groupby('month').agg(np.mean)[['Arithmetic Mean']].rename(co
         aqi = epa_data_CA.get('daily_county_aqi').groupby('Month').agg(np.mean)[['AQI']]

         chemical_concentration_CA = pd.concat([aqi, ozone, so2, co, no2], axis=1, join="inner")


         # PCA Analysis
         chemical_concentration_mean = np.mean(chemical_concentration_CA, axis=0)
         chemical_concentration_std = np.std(chemical_concentration_CA, axis=0)
         chemical_concentration_centered = chemical_concentration_CA - chemical_concentration_mean
         chemical_concentration_normalized = chemical_concentration_centered / chemical_concentration_st

         u, s, vt = np.linalg.svd(chemical_concentration_normalized, full_matrices = False)

         total_variance_computed_from_data = sum(np.var(chemical_concentration_CA, axis=0))

         total_variance_computed_from_singular_values = sum(s**2/chemical_concentration_normalized.shape

         variance_explained_by_1st_pc = sum((s[:1]**2/chemical_concentration_normalized.shape[0])/total_

         chemical_concentration_CA_pcs = pd.DataFrame(u @ np.diag(s)).loc[:,0:1].rename(columns={
             0: 'pc1', 1: 'pc2'})


         # temperature data linear regression analysis
         import matplotlib.pyplot as plt

         data_temp = epa_data_CA.get('daily_wind')
         data_temp['Defining Site'] = data_temp['State Code'].astype(str).str.zfill(2) + '-' + data_temp

         temp_CA = pd.merge(epa_data_CA.get('daily_county_aqi').drop_duplicates()[['Defining Site', 'AQ
         temp_CA = temp_CA[temp_CA['Arithmetic Mean'] < 105]
         temp_CA = temp_CA.dropna()
         temp_CA = temp_CA[temp_CA['AQI'] <= 500]
         temp_CA['temp buckets'] = pd.cut(temp_CA['Arithmetic Mean'], 55).apply(lambda x: int(x.left))
         data_bucket = temp_CA.groupby('temp buckets').agg(np.mean)

         #data_bucket.head()

         chemical_concentration_CA_pcs

         # Will have to normalize data
         # also account for observation count
```

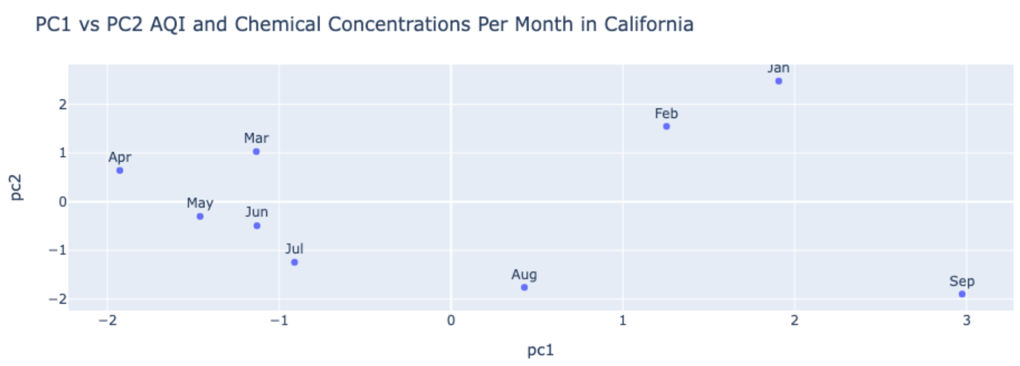
Out[29]:

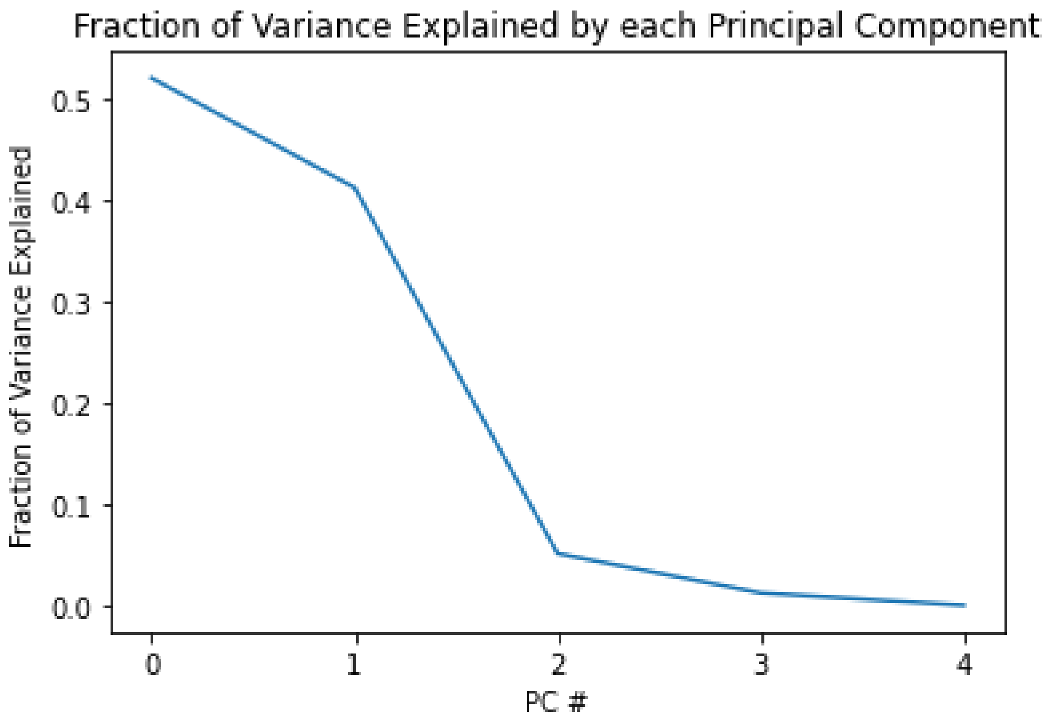
```
        pc1       pc2
0  1.906861  2.478950
1  1.253905  1.547628
2 -1.133341  1.030234
3 -1.927566  0.641893
4 -1.460497 -0.302213
5 -1.129421 -0.494116
6 -0.910867 -1.244532
7  0.426860 -1.759939
8  2.974067 -1.897906
```

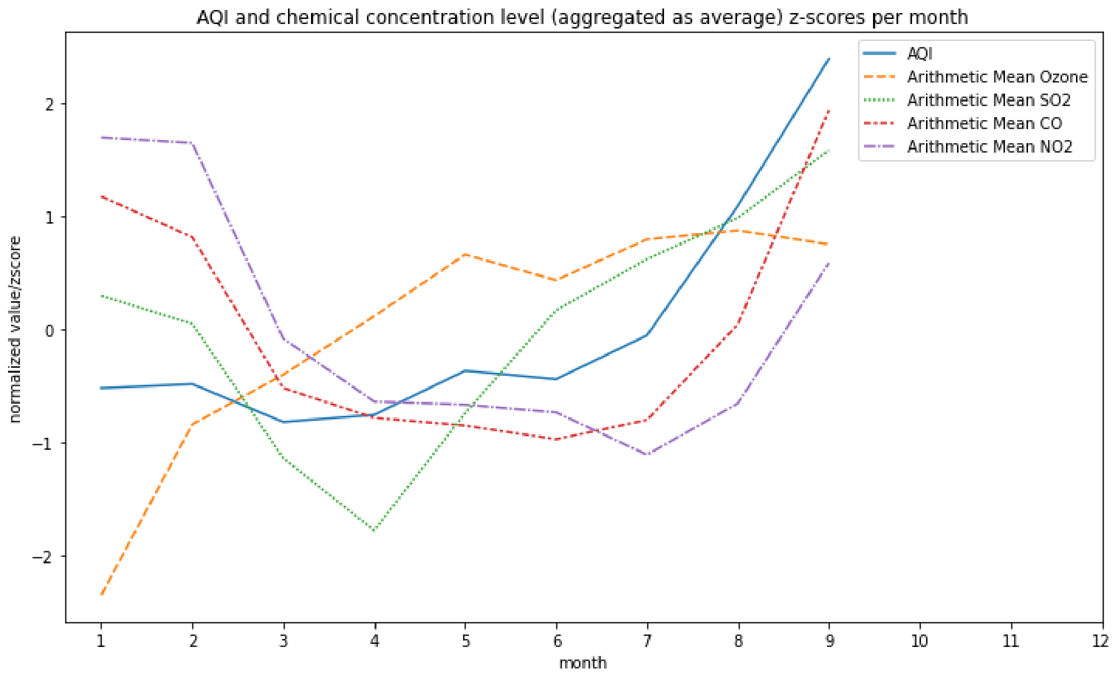
0.0.7 Question 7b - Visualization

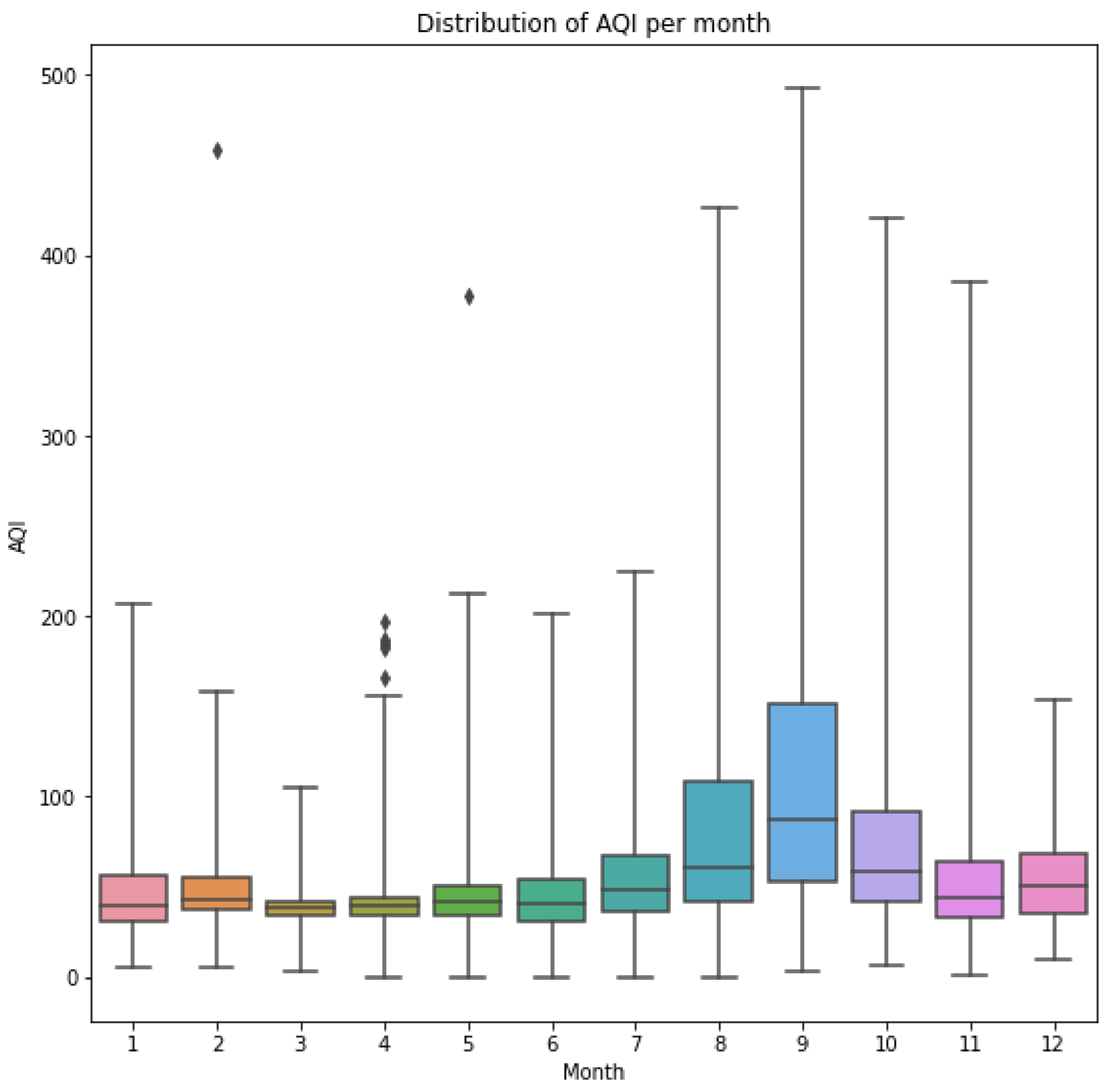
Please create **two** visualizations to summarize your analysis above. The only restrictions are that these visualizations **cannot** simply be scatterplots between two features in the dataset(s) and **cannot** be of the same type (dont make two bar graphs, for example).

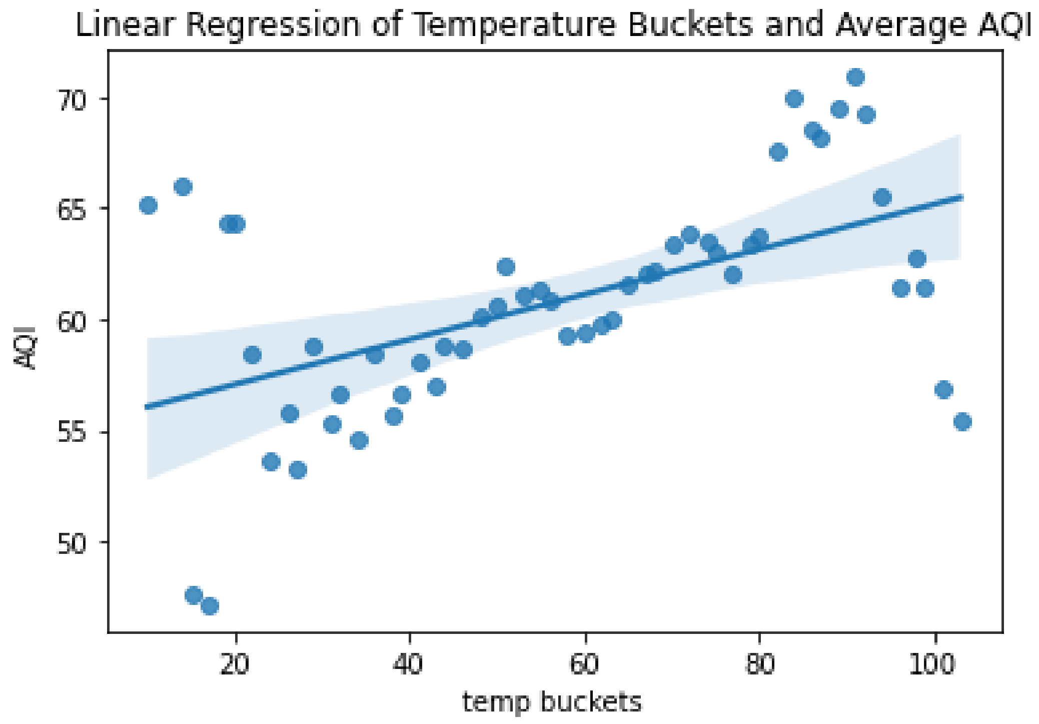
```
In [30]: #plt.scatter(...)

         plt.plot(s**2 / sum(s**2))
         plt.xticks(np.arange(5), np.arange(5))
         plt.xlabel('PC #')
         plt.ylabel('Fraction of Variance Explained')
         plt.title('Fraction of Variance Explained by each Principal Component');

         import plotly.express as px

         county_names = list(chemical_concentration_CA.index)
         chemical_concentration_CA_pcs['Month'] = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Au
         fig = px.scatter(chemical_concentration_CA_pcs, x="pc1", y="pc2", text='Month');
         fig.update_traces(textposition='top center');
         fig.update_layout(title='PC1 vs PC2 AQI and Chemical Concentrations Per Month in California');
         fig.show();

         plt.figure(figsize=(12,7))
         epa_data_CA.get('daily_ozone')[['AQI','Arithmetic Mean']]
         sns.lineplot(data=chemical_concentration_normalized)

         #add title, axes labels
         plt.xticks([1,2,3,4,5,6,7,8,9,10,11,12])
         plt.xlabel('month')
         plt.ylabel('normalized value/zscore')
         plt.title('AQI and chemical concentration level (aggregated as average) z-scores per month');

         #whisker plots showing aqi by month
         t = epa_data_CA.get('daily_county_aqi')
         t = t[t['AQI']<500]
         plt.figure(figsize=(9,9))
         sns.boxplot(x="Month",y="AQI",data=t, whis=12)
         plt.title('Distribution of AQI per month')
         plt.show();

         sns.regplot(data=data_bucket, x = data_bucket.index, y='AQI')
         plt.title('Linear Regression of Temperature Buckets and Average AQI')
         plt.show();
```


PC1 vs PC2 AQI and Chemical Concentrations Per Month in California
pc2
pc1
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
2
1
0
−1
−2
−2
−1
0
1
2
3


Fraction of Variance Explained by each Principal Component
Fraction of Variance Explained
PC #
0.5
0.4
0.3
0.2
0.1
0.0
0
1
2
3
4


AQI and chemical concentration level (aggregated as average) z-scores per month
AQI
Arithmetic Mean Ozone
Arithmetic Mean SO2
Arithmetic Mean CO
Arithmetic Mean NO2
normalized value/zscore
month
2
1
0
−1
−2
1
2
3
4
5
6
7
8
9
10
11
12


Distribution of AQI per month
AQI
0
100
200
300
400
500
Month
1
2
3
4
5
6
7
8
9
10
11
12


Linear Regression of Temperature Buckets and Average AQI
AQI
70
65
60
55
50
20
40
60
80
100
temp buckets

0.0.8 Question 7c - Summary

In a paragraph, summarize the your findings and visualizations and explain how they will be useful for predicting AQI. Make sure that your answer is thoughtful and detailed in that it describes what you did and how you reached your conclusion.

For our open ended EDA, we decided to run a principal component analysis utilizing chemical concentrations as our components. Specifically we focused on the chemical concentrations of ozone, SO2, CO, and NO2. Looking through our clusters, we could clearly see that large values for PC1 represent the months of fall and small values for PC1 represent the warmer temperatures. We also notice that the colder months seem to have larger SO2, CO, and NO2 values. PC2 was slightly more difficult to interpret in this case as the clusters were not very clearly defined. Our guess is that larger PC2 values are associated with larger AQIs and respectively smaller PC2 values with lower AQI levels. The scree plot was used as a form of visualization for the magnitude of variability associated with the individual components obtained from performing PCA. From this plot, we were able to determine the results from our analysis were useful because the scree plot shows that PC1 and PC2 capture more than 80 % of all variance, so therefore we are able to ignore the rest of the PC's without losing any valuable information. To further study the pattern that we noticed of rising chemical concentrations and AQI during colder months we plotted the line graph using the normalized dataset from the PCA analysis to demonstrate that there does indeed seem to be an increase of AQI and and chemical concentrations in colder climate months. From creating our multiple whisker plots, we decided to cap AQI's at 500 for the purposes of any future analyses to ensure that we are not affected by these extreme outlier values. A 500 AQI already signals as hazardous as possible so after making this change we get a better distribution on our whisker plots. Before capping, we were seeing AQI values as high as 1500 and greater and felt that it would be best to remove these outliers to best represent our data.Then, we decided to conduct some analysis on temperature and AQI that could possibly assist us in modeling and predicting AQI for part 2 of this project. We created temperature buckets and ran a simple linear regression model. The resulting line plot showed a strong positive linear association between AQI and temperature. Thus, we believe that temperature and chemical concentrations can be important features in predicting AQI that we will explore further in part 2.

0.0.9 Question 8c - Other Interpolation Ideas

Instead of just interpolating in a simple fashion as we did above, suggest one other way to interpolate (that actually works so do not just say "put the average of all cells in every `NaN` cell). For example, you can take into account of the distance of the surrounding cells, the number of cells you use, and more.

Instead of interpolating as done above, we could gather the average of a range of values that are near the surrounding cell rather than just the closest four we are able to find. It may be beneficial to do the closest two or three for right, left, above, and below and interpolate by the average of all values so that we can try to reduce the number of Nan values we are gathering at the moment. Another alternative could be to take all values from the given latitude and longitude buckets and average those. Additionally, we can place weights on the data based on distance. Data points that are deemed "far" from the NaN data point can have lower weights than data closer to the NaN data point. We can then normalize the average of these data points and find more accurate values that are based on distance.

0.0.10 Question 9 - Choosing your Loss Function

Let us say that you are trying to define a loss function $L(x_i, y_i)$ to use for model, where x_i is the input and the y_i is a qualitative variable that that model outputs, consisting of the following five groups: good, moderate, unhealthy for sensitive groups, unhealthy, very unhealthy, or hazardous. How would you design your loss function to evaluate your model?

Ideally, the loss function would quantitatively measure the distance between the actual and predicted AQI categorical value. The approach would be similar to that of the RMSE and in order to be able to utilize this, a transformation of the categorical AQI variables to numerical standards would occur: good = 1, moderate = 2, unhealthy for sensitive groups = 3, unhealthy = 4, very unhealthy = 5, and hazardous = 6. Then, the residual would be calculated between the actual and predicted categories to represent how far away the prediction was from the actual output and penalize far away values by squaring them, doing this for each prediction, dividing by n to get the average, and taking the square root to get an interpolable value. The goal would be to get a low 'rmse' value because it means the predictions were closer to the actual value.

0.0.11 Question 10a: Choose Features and Model

First, decide on the features that you will be using for your model. How predictive do you think each of the features that you chose will be of the AQI category? Then, how will you choose to make your model (multiple regression, decision trees, etc.)?

The features utilized for the model include the average wind per latitude and longitude location, average ozone concentration per latitude and longitude location, the month the AQI measurement was taken, the latitude and longitude location, as well as total confirmed COVID cases per month. Wind and ozone were features we also thought could be predictive for AQI from part 1 of this project and design document. We believe wind affects AQI directly as it impacts the amount of particulates in the air. We believe month will be an extremely predictive feature from our initial open-ended EDA because it seemed that certain months had much higher AQI values like September for example, most likely due to wildfires. We believe latitude and longitude are predictive because certain locations may be more prone to wildfires and thus raising AQI. Finally, we believe total confirmed covid cases per month can be very predictive because they are directly related to stay-at-home orders and times when traffic significantly decreased due to the number of confirmed covid cases and hence may cause a decrease in AQI. Since the goal for this section is to predict AQI categories, we went with the random forest model. We initially chose a decision tree as it's clear there are six different decisions that could be made, and since each feature is related to the AQI category in different ways (positive association vs negative, etc) a decision tree would provide ease and accurate results without having to find ways to transform the data or remove features we think are important. We realized one decision tree model itself may be overfitting or not as predictive and thus a random forest model would be the most accurate choice as it can aggregate various decision tree models into one.

Since confirmed COVID cases per month was not utilizable due to gradescope issues, we switched over to utilzie traffic data and believe this is still a significant feature because traffic is related to release of particulates in the air which we determined above is correlated with AQI.

0.0.12 Question 11a

Train a baseline model of your choice using any supervised learning approach we have studied to answer your hypothesis and predict something related to AQI; you are not limited to a linear model. However, you may use a maximum of **three features** for this part. After training, evaluate it on some validation data that you hold out yourself.

```
In [44]: co_ca_m = epa_data_CA.get('daily_co').drop(['Datum','Parameter Name','Sample Duration',
                                                      'Observation Count','Observation Percent',
                                                      '1st Max Value', '1st Max Hour','Method Code',
                                                      'Method Name','CBSA Name','Date of Last Change
         temp_ca_m = epa_data_CA.get('daily_wind').drop(['Datum','Sample Duration','Units of Measure','
                                                       'Observation Count','Observation Percent','1st Ma
                                                       '1st Max Hour','Method Code','Method Name','CBSA 
                                                       'Date of Last Change'],axis=1)

         aqi_ca_m = epa_data_CA_merged[['Month','AQI','Defining Site']]
         temp_ca_m = temp_ca_m.groupby('Defining Site').mean().reset_index()[['Defining Site','County C

         #merging aqi values with temperature
         three_features = aqi_ca_m.merge(temp_ca_m, how = 'left', on='Defining Site',suffixes=('_aqi','

         #setting random seed to get consistent error values
         np.random.seed(100)
         co_ca_m['Defining Site'] = co_ca_m['State Code'].astype(str).str.zfill(2) + '-' + co_ca_m['Cou

         co_ca_m = co_ca_m.drop(['State Code', 'Latitude' , 'Longitude'], axis=1)

         co_ca_m = co_ca_m.groupby('Defining Site').mean().reset_index()

         #merging our aqi, temp, and month dataframe with carbon monixide concentrations and dropping e
         three_features = three_features.merge(co_ca_m, how = 'left', on='Defining Site', suffixes=('_t
         three_features = three_features[three_features['AQI'] <= 500].drop_duplicates()

         #t=temp, co=carbon monoxide
         features_m = three_features[['Month', 'Arithmetic Mean_t', 'Arithmetic Mean_co']]
         targets_m = three_features[['AQI']]
         from sklearn.model_selection import train_test_split
         X_train_m, X_test_m, Y_train_m, Y_test_m = train_test_split(features_m, targets_m, train_size=

         from sklearn import linear_model as lm
         model_m = lm.LinearRegression(fit_intercept=True)
         model_m.fit(X_train_m, Y_train_m)

         model_m.score(X_train_m, Y_train_m)

         def rmse(actual_y, predicted_y):
             return np.sqrt(np.mean((actual_y - predicted_y) ** 2))

         from sklearn.model_selection import KFold
         def compute_CV_error(model, X_train, Y_train):
             kf = KFold(n_splits=5)
```

```
        validation_errors = []

        for train_idx, valid_idx in kf.split(X_train):
            # split the data
            split_X_train, split_X_valid = X_train.iloc[train_idx], X_train.iloc[valid_idx]
            split_Y_train, split_Y_valid = Y_train.iloc[train_idx], Y_train.iloc[valid_idx]

            # Fit the model on the training split
            model.fit(split_X_train, split_Y_train)

            # Compute the RMSE on the validation split
            error = rmse(split_Y_valid, model.predict(split_X_valid))
            validation_errors.append(error)

        return np.mean(validation_errors)

    cv_error_m = compute_CV_error(model_m, X_train_m, Y_train_m)
    rmsetrain = rmse(Y_train_m,model_m.predict(X_train_m))
    print('The accuracy on the training data is', model_m.score(X_train_m, Y_train_m))
    print('The accuracy on the test data is', model_m.score(X_test_m, Y_test_m))
    print('The cross validation error is', cv_error_m)
    print('The train RMSE is', rmsetrain[0])
    print('The test RMSE is', rmse(Y_test_m,model_m.predict(X_test_m))[0])
    print('The R-Squared value is', (np.var(model_m.predict(X_train_m)) / np.var(Y_train_m))[0])


    #--------------------plot code for final report--------------------#

    #---------cross val predicted vs actual line plot ------------#
    #from sklearn.model_selection import cross_val_predict
    #from sklearn import linear_model
    #lr = linear_model.LinearRegression()
    #X,y=features_m, targets_m
    #predicted=cross_val_predict(lr,X,y,cv=5)
    #fig, ax = plt.subplots()
    #ax.scatter(y, predicted, edgecolors=(0,0,0))
    #ax.plot([y.min(), y.max()], [y.min(), y.max()], "k--", lw=4)
    #ax.set_xlabel('Actual AQI')
    #ax.set_ylabel('Predicted AQI')
    #ax.set_title("Cross validation predictions vs actual AQI values")
    #plt.show()

    #--------------residual plot----------------#
    #predvals2 = model_m.predict(X_train_m)
    #residuals2 = Y_train_m - predvals2
    #plt.scatter(predvals2, residuals2)
    #plt.axhline(y=0, color='r', linestyle = '-')
    #plt.xlabel('fitted values')
    #plt.ylabel('residuals')
    #plt.title('residual plot')
```

```
The accuracy on the training data is 0.1492014185756294
The accuracy on the test data is 0.10258610932268697
The cross validation error is 39.08552871980819
The train RMSE is 39.05090544723092
The test RMSE is 37.812361144830355
The R-Squared value is 0.14536926599431868
```

0.0.13 Question 11b

Explain and summarize the model that you used. In your summary, make sure to include the model description, the inputs, the outputs, as well as the cross-validation error. Additionally, talk a little bit about what you would change to your baseline model to improve it. The expected length of your summary should be 8-12 sentences.

Through our initial EDA, we hypothesized that AQI levels are positively correlated with increasing chemical concentrations of SO2, CO, and NO2 and increasing temperature. We decided to use only one chemical concentration as it was shared in question 10 of the project that AQI is directly based on chemical concentrations so there would be too high of a correlation. We utilized linear regression as our baseline model. The following three features/inputs were used to discover the type of relationship they have for our initial model analysis: average carbon monoxide concentration by defining site, average temperature by defining site, and the month in which the data was collected. Our output/target are AQI values. From running our analysis, we obtained a training accuracy of 11.33% and a CV error of approximately 39.23. This score implied that our first model did not have an accurate prediction of AQI with the three features used. It follows that there does not seem to be a linear relationship between our selected features and AQI values. We believe that we need to add one more chemical concentration feature to our model as we discovered from our initial EDA that chemical concentrations are positively correlated with AQI and so adding one more can drastically improve our model. Since we cannot add all chemical concentrations, we believe traffic may be a good indicator due to pollutants released, as well as precipitation due to its decrease in pollutants. We may also drop any duplicates in our data and remove outlier values for AQI.

0.0.14 Question 11c

Improve your model from part 11a based on the improvements that you suggested in part 11b. This could be the addition of more features, performing additional transformations on your features, increasing/decreasing the complexity of the model itself, or really anything else. You have no limitation on the number of features you can use, but you are required to use at least **one external dataset** that you process and merge in yourself.

External dataset for question 11c: The precipitation data was obtained from the National Centers for Environmental Information for the timeline of January - December 2020 through this link: https://www.ncdc.noaa.gov/cag/county/mapping/4/pcp/202012/12/value

This link should take you to the exact year of data acquired; however, if it does not, you may county map with the following options

State: California Parameter: Precipitation Year: 2020 Month: December Time Scale: 12-Month

From there, scroll down to the table and there should be a 'Download Table Data:' section were you should click the green icon to download the .csv file.

After the file has been downloaded, open the file on excel and delete the first three rows which contain the phrases:

'January - December 2020 California County Precipitation', 'Units: Inches', 'Missing: -99', and 'Rank out of 126 Years'.

Once these are deleted, save the changes.

Now drag and drop the file, which should be named '4-pcp-202012-12.csv' under the AQI_Part2 project folder. This should be the main folder that contains the data, images, and tests folders along with the AQI.ipynb.

After this, the code should read the file and produce the tables when 11c in run.

```
In [45]: so2_ca_m = epa_data_CA.get('daily_so2').drop(['Datum','Parameter Name','Sample Duration', 'Obs
                                                       'Method Name','CBSA Name','Date of Last Change

         so2_ca_m = so2_ca_m[['County Name', 'State Code', 'County Code','Site Num','Arithmetic Mean']]
         np.random.seed(100)

         so2_ca_m['Defining Site'] = so2_ca_m['State Code'].astype(str).str.zfill(2) + '-' + so2_ca_m['

         so2_ca_m = so2_ca_m.drop(['State Code', 'County Code', 'Site Num'], axis=1)
```

```
three_features = three_features.groupby('Defining Site').mean().reset_index()
multiple_features = three_features.merge(so2_ca_m, how = 'left', on='Defining Site', suffixes=

data_m = traffic_data[['County', 'Back_AADT', 'Ahead_AADT', 'Lon_S_or_W', 'Lat_S_or_W']]
traffic_data_cleaned_m = data_m.replace(r'^\s*$', 0, regex=True)
traffic_data_cleaned_m['AADT'] = traffic_data_cleaned_m['Back_AADT'].astype(int) + traffic_data
traffic_data_cleaned_m.drop(columns = ['Back_AADT', 'Ahead_AADT'], inplace=True)
traffic_data_cleaned_m.rename(columns={'Lon_S_or_W' : 'Longitude', 'Lat_S_or_W' : 'Latitude'},
traffic_data_cleaned_m['Latitude'] = pd.to_numeric(traffic_data_cleaned_m['Latitude'], errors=
traffic_data_cleaned_m['Longitude'] = pd.to_numeric(traffic_data_cleaned_m['Longitude'], errors

traffic_pt_m = gpd.points_from_xy(traffic_data_cleaned_m.Longitude, traffic_data_cleaned_m.Lat
geo_traffic_m = gpd.GeoDataFrame(traffic_data_cleaned_m, geometry=traffic_pt_m)
geo_traffic_m = geo_traffic_m.rename(columns={"Longitude" : "Traffic Long", "Latitude": "Traff

epa_pt_m = gpd.points_from_xy(epa_data_CA_merged.Longitude, epa_data_CA_merged.Latitude)
geo_epa_m = gpd.GeoDataFrame(epa_data_CA_merged, geometry=epa_pt_m)
geo_epa_m = geo_epa.rename(columns={"Longitude" : "Site Long", "Latitude": "Site Lat"})

gpd_epa_traffic_m = gpd.sjoin_nearest(geo_epa_m, geo_traffic_m)

gpd_epa_traffic_m.drop(columns = ['State Name', 'Day', 'Category', 'Site Lat', 'Site Long', 'g

multiple_features = multiple_features.groupby('Defining Site').mean().reset_index()

multiple_features = multiple_features.merge(gpd_epa_traffic_m, how = 'left', on='Defining Site
multiple_features.drop(columns = ['Site Num_co', 'County Code_co', 'Site Num_t'], inplace=True
multiple_features['County Code_t'] = multiple_features['County Code_t'].astype(int).astype(str
multiple_features.rename(columns={'County Code_t' : 'County Code'}, inplace=True)

#-------------------------------------EXTERNAL DATA---------------------------------------#
precipitation_data = pd.read_csv('4-pcp-202012-12.csv')[['Location ID', 'Value']]
precipitation_data['Location ID'] = precipitation_data['Location ID'].str.replace(r'\D', '')
precipitation_data.rename(columns={'Location ID' : 'County Code', 'Value': 'Precipitation Avera
multiple_features = multiple_features.merge(precipitation_data, how = 'left', on='County Code'

#t=temp, co=carbon monoxide, arithmetic mean = so2, aadt = traffic
features_m_2 = multiple_features[['Month', 'Arithmetic Mean_t', 'Arithmetic Mean_co', 'Arithme
targets_m_2 = multiple_features[['AQI']]

X_train_m_2, X_test_m_2, Y_train_m_2, Y_test_m_2 = train_test_split(features_m_2, targets_m_2,

model_m_2 = lm.LinearRegression(fit_intercept=True)
model_m_2.fit(X_train_m_2, Y_train_m_2)

model_m_2.score(X_train_m_2, Y_train_m_2)

cv_error_m_2 = compute_CV_error(model_m_2, X_train_m_2, Y_train_m_2)

print('The accuracy on the training data is', model_m_2.score(X_train_m_2, Y_train_m_2))
print('The accuracy on the test data is', model_m_2.score(X_test_m_2, Y_test_m_2))
print('The cross validation error is', cv_error_m_2)
print('The train RMSE is', rmse(Y_train_m_2,model_m_2.predict(X_train_m_2))[0])
print('The test RMSE is', rmse(Y_test_m_2,model_m_2.predict(X_test_m_2))[0])
print('The R-Squared value is', (np.var(model_m_2.predict(X_train_m_2)) / np.var(Y_train_m_2))

print('----------------------------------------')
print('Feature Selection')
#feature selection
errors = []
range_of_num_features = range(1, X_train_m_2.shape[1] + 1)
for N in range_of_num_features:
    print(f"Trying first {N} features")
    model = lm.LinearRegression()

    # compute the cross validation error
    error = compute_CV_error(model, X_train_m_2.iloc[:,0:N], Y_train_m_2)

    print("\tCV RMSE:", error)
    errors.append(error)

best_num_features = range_of_num_features[np.argmin(errors)]
best_err = np.min(errors)

print(f"Best choice, use the first {best_num_features} features")

#---------------------plot code for final report--------------------_#
#----------------residual plot with noise------------#
#predvals2 = model_m_2.predict(X_train_m_2)
#residuals2 = Y_train_m_2 - predvals2
#plt.scatter(predvals2+np.random.normal(0,10,size=(585,1)), residuals2+np.random.normal(0,10,s
#plt.axhline(y=0, color='r', linestyle = '-')
#plt.xlabel('fitted values')
#plt.ylabel('residuals')
#plt.title('residual plot')

#---------------residual plot without noise-------------#
#predvals2 = model_m_2.predict(X_train_m_2)
#residuals2 = Y_train_m_2 - predvals2
#plt.scatter(predvals2, residuals2)
#plt.axhline(y=0, color='r', linestyle = '-')
#plt.xlabel('fitted values')
#plt.ylabel('residuals')
#plt.title('residual plot')

#-------------bar plot rmse comparison--------------#
#vallss = [39.05090544723092, 39.08552871980819, 37.812361144830355, 7.846216900510057,7.98122
#modell = ['3 feature', '3 feature', '3 feature', 'improved','improved','improved']
#typee = ['train','cv','test','train','cv','test']
#tbll = pd.DataFrame({'RMSE':vallss,'model':modell,'type':typee})
#sns.barplot(x='model',y='RMSE', data=tbll, hue='type')
#plt.xlabel('model type')
#plt.ylabel('rmse')
#plt.title('rmse by model and type')

#------------cross val predicted vs actual aqi values with noise------------_#
#from sklearn.model_selection import cross_val_predict
#from sklearn import linear_model
```

```
#lr = linear_model.LinearRegression()
#X,y=features_m_2, targets_m_2
#predicted=cross_val_predict(lr,X,y,cv=5)
#fig, ax = plt.subplots()
#ynoise = y + np.random.normal(0,3,size=(837,1))
#prednoise = predicted + np.random.normal(0,3,size=(837,1))
#ax.scatter(ynoise, prednoise, 100, edgecolors=(0,0,0))
#ax.plot([ynoise.min(), ynoise.max()], [ynoise.min(), ynoise.max()], "k--", lw=4)
#ax.set_xlabel('Actual AQI')
#ax.set_ylabel('Predicted AQI')
#ax.set_title("Cross validation predictions vs actual AQI values")
#plt.show()

#-------------------cross val predicted vs actual aqi values no noise----------#
#from sklearn.model_selection import cross_val_predict
#from sklearn import linear_model
#lr = linear_model.LinearRegression()
#X,y=features_m_2, targets_m_2
#predicted=cross_val_predict(lr,X,y,cv=5)
#fig, ax = plt.subplots()
#ynoise = y
#prednoise = predicted
#ax.scatter(ynoise, prednoise, 100, edgecolors=(0,0,0))
#ax.plot([ynoise.min(), ynoise.max()], [ynoise.min(), ynoise.max()], "k--", lw=4)
#ax.set_xlabel('Actual AQI')
#ax.set_ylabel('Predicted AQI')
#ax.set_title("Cross validation predictions vs actual AQI values")
#plt.show()
```

```
The accuracy on the training data is 0.669897170573487
The accuracy on the test data is 0.7105707493719433
The cross validation error is 7.981224187607308
The train RMSE is 7.846216900510057
The test RMSE is 7.02024380598197
The R-Squared value is 0.6574550764308008
--------------------------------------
Feature Selection
Trying first 1 features
        CV RMSE: 13.594035239752378
Trying first 2 features
        CV RMSE: 12.435264256597495
Trying first 3 features
        CV RMSE: 11.431320218680181
Trying first 4 features
        CV RMSE: 11.324695152788774
Trying first 5 features
        CV RMSE: 11.0650666130316
Trying first 6 features
        CV RMSE: 7.981224187607308
Best choice, use the first 6 features
```

0.0.15 Question 11d

Compare and contrast your baseline model and (hopefully) improved model. Make sure to compare their validation errors. Were you able to successfully answer your research question and evaluate your hypothesis? Summarize in a few sentences the conclusions that you can draw from your model and predictions. The expected length of your response should be 8-10 sentences.

We found a drastic increase in our accuracy and decrease in our CV error with our improved model. The training accuracy improved to 68.72% (baseline was 11.33%) and the CV error for the improved model is approximately 7.92 (baseline was 39.23). We also did feature selection and found that CV error was minimized with all six features so no further analysis was needed. I believe we were able to successfully answer our hypothesis because our test accuracy of approximately 66%, means that there does seem to be some linear relationship between AQI, chemical concentrations, and temperature. Though we could not use all chemical concentrations simultaneously, traffic and precipitation ended up being valuable indirect measures of chemical particulates to further help predict AQI levels. It seems that from this model we can draw the conclusion that AQI values are linearly associated with weather, and pollutants. To provide further detail, temperature and precipitation were our weather values. Carbon monoxide, sulfur dioxide, and traffic as an indirect measure of pollution were our variables for pollutants. However, our accuracy was not extremely high so further analysis needs to be done on the impact of these features to further improve the model.

To double-check your work, the cell below will rerun all of the autograder tests.

```
In [46]: grader.check_all()
```

```
Out[46]: q10b results: All test cases passed!

         q10c results: All test cases passed!

         q10d results: All test cases passed!

         q10e results: All test cases passed!

         q1a results: All test cases passed!

         q1b results: All test cases passed!

         q1c results:
             q1c - 1 result:
                 Trying:
                     set(epa_data_CA_merged.columns) == set(['State Name', 'county Name', 'Month', 'Day
                             'Defining Site', 'Latitude', 'Longitude'])
                 Expecting:
                     True
                 **********************************************************************
```

```
        Line 1, in q1c 0
        Failed example:
            set(epa_data_CA_merged.columns) == set(['State Name', 'county Name', 'Month', 'Day
                    'Defining Site', 'Latitude', 'Longitude'])
        Expected:
            True
        Got:
            False

    q1c - 2 result:
        Trying:
            assert epa_data_CA_merged[epa_data_CA_merged['county Name'] == 'Alameda'].shape ==
        Expecting nothing
        **********************************************************************
        Line 1, in q1c 1
        Failed example:
            assert epa_data_CA_merged[epa_data_CA_merged['county Name'] == 'Alameda'].shape ==
        Exception raised:
            Traceback (most recent call last):
              File "/opt/conda/lib/python3.9/doctest.py", line 1336, in __run
                exec(compile(example.source, filename, "single",
              File "<doctest q1c 1[0]>", line 1, in <module>
                assert epa_data_CA_merged[epa_data_CA_merged['county Name'] == 'Alameda'].shap
            AssertionError
        Trying:
            round(epa_data_CA_merged[(epa_data_CA_merged['county Name'] == 'Alameda') & (epa_d
                                 & (epa_data_CA_merged['Day'] == 1)]['Latitude'][0], 2) == 37.74
        Expecting:
            True
        ok

q2a results: All test cases passed!

q2b results: All test cases passed!

q3a results: All test cases passed!

q4a results: All test cases passed!

q4i results: All test cases passed!

q5a results: All test cases passed!

q6a results: All test cases passed!

q6b results: All test cases passed!

q8a results: All test cases passed!

q8b results: All test cases passed!
```

0.1 Submission

Make sure you have run all cells in your notebook in order before running the cell below, so that all images/graphs appear in the output. The cell below will generate a zip file for you to submit. **Please save before exporting!**

```
In [47]: # Save your notebook first, then run this cell to export your submission.
         grader.export()
```

```
<IPython.core.display.HTML object>
```