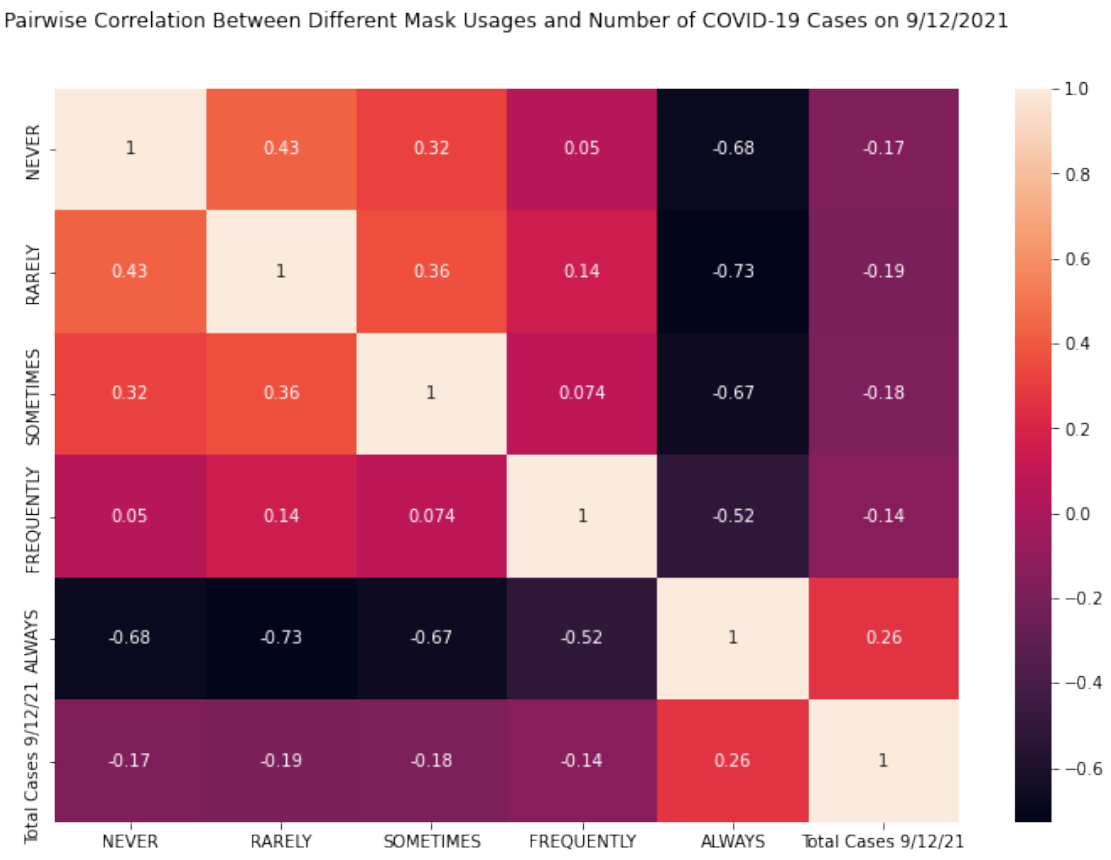
0.0.1 Question 5a

We will use county-wise mask usage data to predict the number of COVID-19 cases on September 12th, 2021. Create a visualization that shows the pairwise correlation between each combination of column in the mask usage data and the number of COVID-19 cases.

Hint: You should be plotting 36 correlations.

```
In [48]: temp5a = county_data.loc[:, "NEVER": "ALWAYS"]
         temp5a["Total Cases 9/12/21"] = county_data["9/12/21"]
         plt.figure(figsize = (12,8))
         q5a_plot = sns.heatmap(temp5a.corr(), annot=True)
         q5a_plot = q5a_plot.set(title = "Pairwise Correlation Between Different Mask Usages and Number
```



0.0.2 Question 5c

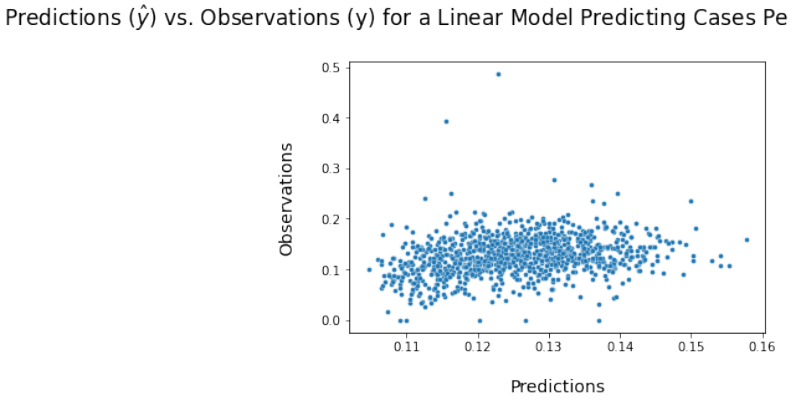
Explain potential reasons the test set RMSE is much higher as compared to the training set RMSE.

There are several potential reasons why the test set RMSE is much higher. The first is that in general, the predictions are based on data the model has never seen before, and the model is fitted based on data that may be different. This creates an inherent risk of overfitting, which would mean we need to find better features. Additionally, the testing data set doesn't have as much data as the training, which potentially increases standard deviation by a significant amount. Finally, different counties have different populations, so using total cases as a response variable isn't the best decision; if 2 counties had wildly different population counts, then having the same total cases mean 2 very different things, which could create a lot of error.

0.0.3 Question 5e

Visualize the model outputs from part (d) by plotting the predictions $\hat{y}$ versus the observations y. Comment on what the plot indicates about our linear model as a comment in the code cell.

```
In [53]: sns.scatterplot(x = lr.predict(X_q5d_test), y = y_q5d_test, s = 15)
         plt.xlabel('\nPredictions', fontsize = 14)
         plt.ylabel('Observations\n', fontsize = 14)
         plt.title('Predictions ($\hat{y}$) vs. Observations (y) for a Linear Model Predicting Cases Pe

         # This plot indicates that for our test data, there are times when our linear model isn't accu
         # Other than that, it appears that the predictions generally are in the same ballpark as obser
```



0.0.4 Question 5h

Using the bias-variance decomposition, comment on how much the variance of the model contributes to the error on the sample point above. We will extend this scenario to analyze the noise term in the bias-variance decomposition, specifically with regards to this COVID-19 dataset. Consider the following:

i) Assuming no observational noise (i.e. $\epsilon = 0$), what is the *magnitude* of the empirical model bias on this sample point?

ii) Clearly, there is a non-trivial amount of observational noise with COVID-19 case data simply due to how testing works and occurs. Please take a look at this article for more information. Given this infomation, explain the issues with the assumptions and result in 5h(i).

iii) Recall that we typically assume $y = g(x) + \epsilon$, where $\epsilon \sim \mathcal{N}(\mu, \sigma)$. In the theoretical setting for bias-variance, we have assumed $\mu = 0, \sigma > 0$. In this practical setting, analyze and determine how ϵ could be modeled (as a normal distribution, but you may also consider how it could be modeled as another distribution). Are there any immediate issues with the assumptions we have made in the theoretical setting where $\mu = 0, \sigma > 0$? What conditions on μ, σ could be more appropriate and why?

iv) Does the standard bias-variance decomposition presented in lecture hold given ϵ being normally distributed according to your answer in part (iii)? If so, please explain why. If not, explain why it does not hold and if possible, how to modify the bias-variance decomposition to make it hold (i.e. perhaps there is an additional noise term $E[\epsilon^3]$).

Hint: Try to express $y = g(x) + \epsilon$ by adding and subtracting a certain quantity.

v) Intuitively, is it possible to minimize the true model bias to 0 given your ϵ formulation in part (iii)? Why or why not? Justify your answer using part (iv) if possible.

vi) Consider the infinite sample case where an oracle grants you as many samples as requested, and answer the same question in part (v). Is it possible to minimize the true model bias to 0 given your ϵ formulation in part (iii)? Conclude with an analysis of what our modeling task can approximate using $X\theta \approx y$ in the finite and infinite sample case.

Student Answer:

i) If there is no observational noise, then the magnitude of the empirical model bias on this sample point is equal to the square root of [error - variance] (after simplifying the formula error = observation variance + model bias^2 + model variance). Although some randomness may be involved, this value should be roughly around 0.04, as seen in the following cell - note that the real value of the bias can be either positive or negative since it is a square root.

```
In [58]: # (i) of Q 5h:
```

```
model_bias = (q5g_error - q5g_var) ** 0.5
model_bias
```

```
Out[58]: 0.04125193229594931
```

ii) There are a few issues with our assumptions of no observational noise. Because of different factors in how COVID interacts with governments and healthcare systems around the world, our model intrinsically doesn't account for arbitrary and unpredictable inconsistencies. For instance, the provided article conveys that the number of cases doesn't truly represent the state of COVID within a given geography; this is because higher cases could be due to either increased rates of cases per capita (a sign that conditions are worsing), or expanded testing capabilities (a good sign reflecting that the healthcare system is becoming less overwhelmed). On the other hand, false positives and false negatives, which occur more frequently with certain types of tests, also influence the gap between the amounts of positive tests and true cases.

iii) We most likely have negative bias, since the model underestimates how many cases there are due to lack of comprehensive testing, as per the article. Thus, when reflecting noise, we should have a negative μ to account for this, as the number of positive tests is less than the true value of COVID cases. σ should remain the same, as there is no indication that it is different.

iv) If ϵ is normally distributed, then $\mathbb{E}(\epsilon)$ will be μ, i.e. the mean noise. Following the logic of the proof at the link below, which is based on the proof presented in lecture, we can see that the standard bias-variance decomposition does indeed hold, as it remains true that model risk = observation variance + model bias^2 + model variance. However, as displayed in the top right of the proof, the model bias term is defined differently. Instead of model bias = $\mathbb{E}(\hat{Y}(x)) - g(x)$, the correct representation is now model bias = $\mathbb{E}(\hat{Y}(x)) - (g(x) + \mu) = \mathbb{E}(\hat{Y}(x)) - g(x) - \mu$ since E(ϵ) no longer equals zero. This distinction is important even though the general decomposition remains the same.

https://tinyurl.com/5h-part-iv

v) It isn't possible to minimize bias to 0 because, based on the answer to the previous part, the model bias term will always have a μ term that is a non-zero value, since ϵ is centered at a non-zero value now.

vi) In the infinite sample case, it may be more possible to model this noise, but we believe that it is still impossible; we will never know the "true" number of COVID cases, which is distinct from positive test results. There is no way to find the distinction between the two, even with infinite data. Models can likely only reliably predict positive test results, and not true cases, given different factors surrounding testing practices.

0.1 Question 5j

Propose a solution to reducing the mean square error using the insights gained from the bias-variance decomposition above. What are the values of the quantities that have we estimated and what can be concluded about the remaining quantities? Please show all work that informs your analysis.

Assume that the standard bias-variance decomposition used in lecture can be applied here.

The mean squared error generally measures the average of the errors squared (difference between estimated values and original values). To reduce the mean squared error, we could attempt to reduce bias / variance by either increasing model complexity or even altering the features included in the model; it should be noted, however, that additional features or overly complex models could unintentionally increase variance as well, which may counteract the original intention of reducing bias and variance. Herein lies the bias-variance trade-off where bias can be easily reduced by increasing the variance, and vice versa, Simpler models (linear / logistic regression) tend to have higher bias and lower variance, while more complex models (e.g. random forest) generate lower bias but higher variance values. Overfitting here remains a risk since variance will increase, so features must be selected so as to reduce additional noise. K-fold cross-validation could be used here to determine the features that generate the lowest mean square error. Choosing a model configuration based on the generated values for bias and variance is key—k in k-nearest neighbors controls bias-variance tradeoffs (smaller values generate lower bias and higher variance whereas larger k values result in higher bias and lower variances).

The average mean squared error we generated came out to approximately 0.00146 while average variance using the bias-variance decomposition came out to 4.1272E-06, indicating that our models work relatively well when evaluating the entire test set from d).

0.1.1 Question 6a

Train a baseline model where f is the model described in Question 0a and x is a quantity of *your* choice. Note that you may used *any* supervised learning approach we have studied; you are not limited to a linear model.

In [11]:
```
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

q6a = pd.DataFrame(data = edited)


#the following arrays represent how many days after the foreign travel ban that x action took


days_since_stay_temp = edited["stay at home"] - edited["foreign travel ban"]
days_since_stay_temp
days_since_stay = [days_since_stay_temp.loc[i].days for i in days_since_stay_temp.index]

days_since_50_temp = edited[">50 gatherings"] - edited["foreign travel ban"]
days_since_50 = [days_since_50_temp.loc[i].days for i in days_since_50_temp.index]

days_since_500_temp = edited[">500 gatherings"] - edited["foreign travel ban"]
days_since_500 = [days_since_500_temp.loc[i].days for i in days_since_500_temp.index]

days_since_school_temp = edited["public schools"] - edited["foreign travel ban"]
days_since_school = [days_since_school_temp.loc[i].days for i in days_since_school_temp.index]

days_since_food_temp = edited["restaurant dine-in"] - edited["foreign travel ban"]
days_since_food = [days_since_food_temp.loc[i].days for i in days_since_food_temp.index]

days_since_e_temp = edited["entertainment/gym"] - edited["foreign travel ban"]
days_since_e = [days_since_stay_temp.loc[i].days for i in days_since_e_temp.index]
```

In [12]:
```
# creating design matrix and further cleaning

X = pd.DataFrame()
X["stay at home"] = days_since_stay
X[">50 gatherings"] = days_since_50
X[">500 gatherings"] = days_since_500
X["public schools"] = days_since_school
X["restaurant dine-in"] = days_since_food
X["entertainment/gym"] = days_since_e
X["dem_to_rep_ratio"] = list(edited["dem_to_rep_ratio"])

X = X[False == (X["dem_to_rep_ratio"].isnull())]
y = np.log(X[["dem_to_rep_ratio"]])
X = X.drop(columns = ["dem_to_rep_ratio"])

for col in list(X):
```

```
            X[col] = X[col].fillna(X[col].max() + 1)


        # train test split
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=42)

        model6a = LinearRegression().fit(X_train, y_train)
        pred6a = model6a.predict(X_test)
        rmse = (mean_squared_error(y_test, pred6a))**0.5
        print("Initial Baseline RMSE: " + str(rmse))
```

```
Initial Baseline RMSE: 0.7462478636175799
```

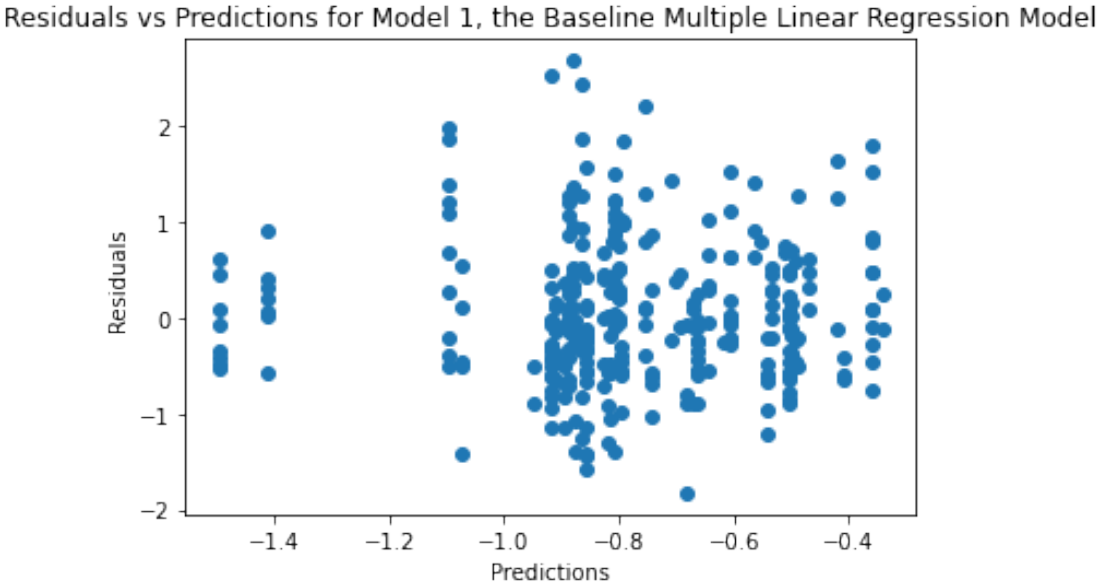
0.1.2 Question 6b

Improve your model from part (a). Specify the supervised model you choose and write $f(x)$ as a function of the chosen features and parameters in your model. Justify why you chose these features and how you expect they will correlate with the output you wish to predict.

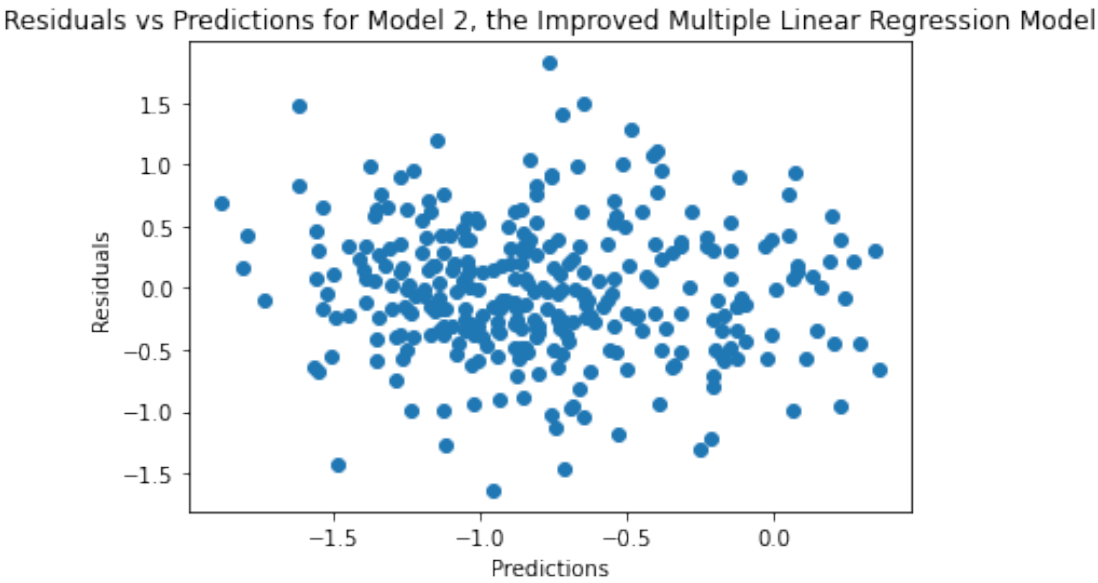
In [13]:
```
#improving multiple linear regression


most_recent_vax = vaccinations[vaccinations["Date"] == vaccinations.sort_values("Date", ascendi
state_pops = county_data.groupby("STNAME").sum()[["POPESTIMATE2020"]].reset_index()
q6bmerged = most_recent_vax.merge(state_pops, left_on = "Province_State", right_on = "STNAME")
q6bmerged["Proportion Fully"] = q6bmerged["People_Fully_Vaccinated"]/q6bmerged["POPESTIMATE202
q6bmerged["Proportion Partially"] = q6bmerged["People_Partially_Vaccinated"]/q6bmerged["POPEST

X6b = pd.DataFrame()

X6b["stay at home"] = days_since_stay
X6b[">50 gatherings"] = days_since_50
X6b[">500 gatherings"] = days_since_500
X6b["public schools"] = days_since_school
X6b["restaurant dine-in"] = days_since_food
X6b["entertainment/gym"] = days_since_e
X6b["dem_to_rep_ratio"] = list(edited["dem_to_rep_ratio"])
X6b["State FIPS"] = list(edited["STATEFP"])
X6b["County FIPS"] = list(edited["COUNTYFP"])
X6b["County FIPS"] = X6b["State FIPS"]*1000 + X6b["County FIPS"]

X6b = X6b[False == (X6b["dem_to_rep_ratio"].isnull())]
X6b = X6b[X6b["dem_to_rep_ratio"] < 22]
X6b = X6b.merge(q6bmerged[["FIPS", "Proportion Fully", "Proportion Partially"]], left_on = "Sta
temp_mask_use = pd.DataFrame(data = mask_use)
temp_mask_use["COUNTYFP"] = [float(temp_mask_use["COUNTYFP"][i]) for i in temp_mask_use.index]
X6b = X6b.merge(temp_mask_use[["COUNTYFP", "NEVER", "ALWAYS"]], left_on = "County FIPS", right_


X6b = X6b.sample(frac = 1, random_state = 44)
y6b = np.log(X6b["dem_to_rep_ratio"])
X6b = X6b.drop(columns = ["dem_to_rep_ratio", "State FIPS", "County FIPS", "FIPS", "COUNTYFP"])

for col in list(X6b):
    X6b[col] = X6b[col].fillna(X6b[col].max() + 1)


# train test split
X6b_train, X6b_test, y6b_train, y6b_test = train_test_split(X6b, y6b, test_size=0.1, random_sta

model6b = LinearRegression().fit(X6b_train, y6b_train)
pred6b = model6b.predict(X6b_test)
rmse6b = (mean_squared_error(y6b_test, pred6b))**0.5
print("Improved RMSE: " + str(rmse6b))
```

```
Improved RMSE: 0.5427378151345704
```

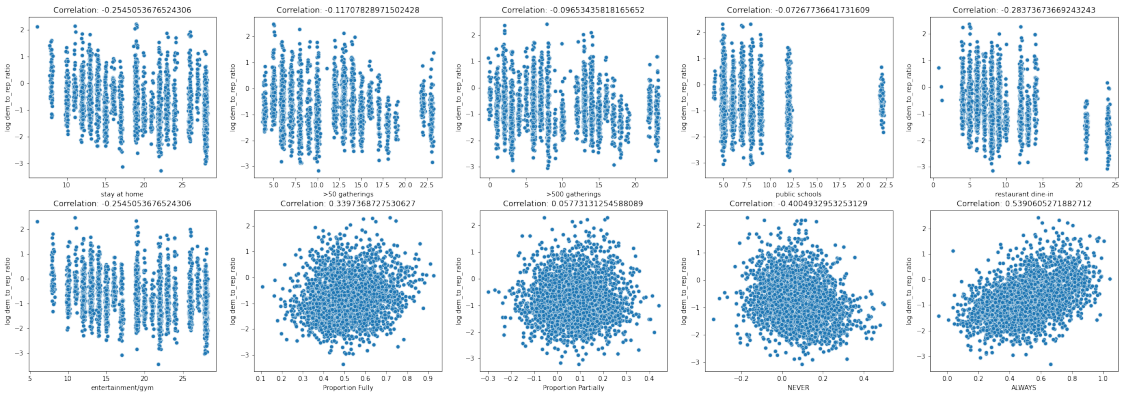
In [31]:
```
# scatter plot for model 1, predictions vs residuals
plt.scatter(x = pred6a, y = y_test-pred6a)
plt.xlabel("Predictions")
plt.ylabel("Residuals")
plt.title("Residuals vs Predictions for Model 1, the Baseline Multiple Linear Regression Model
```



In [30]:
```
# scatter plot for model 2, predictions vs residuals
plt.scatter(x = pred6b, y = y6b_test-pred6b)
plt.xlabel("Predictions")
plt.ylabel("Residuals")
plt.title("Residuals vs Predictions for Model 2, the Improved Multiple Linear Regression Model
```



In [14]:
```
# correlations for multiple linear regression

fig, axes = plt.subplots(2,5, figsize = (30,10))
r = 0
c = 0
for i in list(X6b):

    jitx = X6b[i] + np.random.normal(0, 0.1, size = (len(X6b[i])))
    jity = y6b + np.random.normal(0, 0.1, size = (len(y6b)))


    sns.scatterplot(ax=axes[r,c], x = pd.Series(jitx), y = pd.Series(jity))
    axes[r,c].set_title("\nCorrelation: " + str(np.corrcoef(X6b[i], y6b)[0][1]))
    axes[r,c].set_ylabel("log dem_to_rep_ratio")

    c += 1
    if (c == 5):
        c = 0
        r += 1
```



In [15]: *# cleaner linear*

```
lastX_train, lastX_test, lastY_train, lastY_test = train_test_split(X6b[["ALWAYS", "restaurant
final = LinearRegression().fit(lastX_train, lastY_train)
finalpred = final.predict(lastX_test)
rmsefinal = (mean_squared_error(finalpred, lastY_test))**0.5
print("Final RMSE: " + str(rmsefinal))
```

```
Final RMSE: 0.557265767438431
```

In [68]: *# logistic*

```
edited['log_demrep_ratio'] = np.log(edited['dem_to_rep_ratio'])

log_X = edited[['stay at home', '>50 gatherings',
                '>500 gatherings', 'public schools', 'restaurant dine-in',
                'entertainment/gym']]

log_X = pd.DataFrame()

log_X["stay at home"] = days_since_stay
log_X[">50 gatherings"] = days_since_50
log_X[">500 gatherings"] = days_since_500
log_X["public schools"] = days_since_school
log_X["restaurant dine-in"] = days_since_food
log_X["entertainment/gym"] = days_since_e

from sklearn.preprocessing import binarize
binary_demrep = edited['log_demrep_ratio'].values.reshape(1,-1)
x = binarize(binary_demrep)
edited['log_demrep_ratio'] = np.ravel(binarize(binary_demrep.reshape(1,-1), 0.5))
```

```
        for col in list(log_X):
            log_X[col] = log_X[col].fillna(log_X[col].max() + 1)

        log_Y = pd.Series(edited['log_demrep_ratio'])
        log_X_train, log_X_test, log_Y_train, log_Y_test = train_test_split(log_X, log_Y, test_size=0.
        log_model = LogisticRegression()
        log_X_train

        log_model = log_model.fit(log_X_train, log_Y_train)
        log_Y_pred = log_model.predict(log_X_train)
        print('Logistic Regression Score: ', log_model.score(log_X_test, log_Y_test))
```

```
Logistic Regression Score:  0.9423076923076923
```

In [69]:
```
        # Random Forest

        from sklearn.ensemble import RandomForestClassifier

        rf = RandomForestClassifier()
        rf_model = rf.fit(log_X_train, log_Y_train)


        predicted = pd.DataFrame(rf_model.predict(log_X_test))
        probs = pd.DataFrame(rf_model.predict_proba(log_X_test))
        from sklearn import metrics
        from sklearn.model_selection import cross_val_score
        rf_accuracy = metrics.accuracy_score(log_Y_test, predicted)
        rf_roc_auc = metrics.roc_auc_score(log_Y_test, probs[1])
        rf_precision = metrics.precision_score(log_Y_test, predicted, pos_label=1)
        rf_recall = metrics.recall_score(log_Y_test, predicted, pos_label=1)


        # 10-fold cross-validation
        rf_cv_scores = cross_val_score(rf_model, log_X_test, log_Y_test, cv=10)


        rf_cv_mean = np.mean(rf_cv_scores)
        print('10-fold Cross Validated Score: ' + str(rf_cv_mean))
```

```
10-fold Cross Validated Score: 0.9327620967741936
```

0.1.3 Question 6c

If applicable, write an equation or rule for the prediction function $f(x)$; if this is infeasible, make sure to visualize your model parameters in some way. Interpret your improved model's optimal parameters (*hint*: refer to 1aiii), and compare these parameters to those of the baseline model. Comment on whether the parameters follow physical intuition given the nature of the prediction task.

For example, if you chose to use a decision tree, you may interpret the generated rules.

Type your answer here, replacing this text.

0.1.4 Question 6d

Discuss your improved model's performance on both short-term and long-term time scales using a metric of your choice (*hint:* we're using an autoregressive model). In other words, given x_t, we wish to predict $\hat{x}_{t+k}$, and plot the error of these predictions for two k values of your choice. You may use any reasonable interpretation of short-term and long-term predictions; an initial suggestion is to use 2-day predictions and 2-week predictions.

Compare the performance of this model on both timescales with the baseline model.

```
In [70]: ...
```

```
Out[70]: Ellipsis
```

0.1.5 Question 6e

Plot and describe the error for both the baseline and improved models as a function of time. In other words, given x_t, we wish to predict $\hat{x}_{t+k}$, and plot the error of these predictions for all k.

Consider how and why the performance of the model degrades as a function of time using the rate of growth in the error.

```
In [71]: ...
```

```
Out[71]: Ellipsis
```

0.1.6 Question 6f

Consider a modification to the model $f(x) = x_{t+1}$ where instead $f(x) = [x_{t+1}, x_{t+2}, ..., x_{t+m}]$ for some $m > 1$. In other words, using the features x that contain past and present quantities, our model *explicitly* predicts values for m days in the future rather than simply the next day (i.e. $m = 1$).

Train the baseline and improved model using $m = 5$ and $m = 10$. Evaluate and visualize the predictive accuracy of both models.

```
In [72]: ...
```

```
Out[72]: Ellipsis
```

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

```
In [73]: grader.check_all()
```

```
Out[73]: q1a results: All test cases passed!

         q1d results: All test cases passed!

         q1e results: All test cases passed!

         q5b results: All test cases passed!

         q5d results: All test cases passed!

         q5f results: All test cases passed!

         q5g results: All test cases passed!

         q5i results: All test cases passed!
```

0.2 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 [32]:
```
# Save your notebook first, then run this cell to export your submission.
grader.export()
```

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