Influenza Influences: Prediction of Influenza Vaccine Distribution Using Search Engine Query Data
G2 - The InFLUenzers

Data Science Capstone Project
Predictive Modeling Report

Date:

August 26th, 2022

Team Members:

Name: Jackie Glosson

Name: Tien Nguyen

Name: Ashley Wheeler

Name: Andrew Chen

1. **Define the Predictive Modeling Problem**

A. Input: What are the input data and define the input data clearly?

The data input for our time series algorithm is our google trends “Flu shot” query. This column was the most correlated (.89) to flu shot distribution given our EDA last semester, therefore it will be the only input for our time series algorithm. Input data is all data in the column, output data will be 10 rows per state (one for each month of flu season, so 100 rows in total) corresponding to the google trends prediction for flu season 2022 - 2023

For our algorithm to predict actual vaccine distribution, we will use the following columns. Note that the column FLU SHOT will first be predicted in the time series algorithm, and then that data that we predict for flu season 2022-2023 will be used as input to our algorithm.

- FLU SHOT
- PERCENT_OVER_65
- MEDIAN_HOUSEHOLD_INCOME
- PERCENT_ATTAIN_BACHELORS
- WHITE
- BLACK
- RACE_OTHER
- HISPANIC
- AVG_TEMP
- MONTH
- YEAR
- PERCENT_VOTED_DEMOCRATIC
- PERCENT_VOTED_REPUBLICAN
- PERCENT_UNINSURED
- PERCENT_PRIVATE_INSURANCE
- PERCENT_PUBLIC_INSURANCE

B. Data Representation: What is the data representation?

Data representation focuses on allowing us to summarize and display the data, like a data dictionary. It helps us to understand the meaning of the data. Data representation aims to answer questions such as: What is the spread of the data? And what is the typical value of the data (whether qualitative or quantitative)?

Attribute	Data Type	Number of NA	Meaning of Attribute
GET FLU SHOT	float	0	Google Trend search term
FLU SHOT	float	0	Google Trend search term
FLU SHOT NEAR ME	float	0	Google Trend search term
FLU VACCINE	float	0	Google Trend search term
FLU SHOT SIDE EFFECTS	float	0	Google Trend search term
PERCENT_OVER_65	float	560	Percent of people who are 65-year-old and above (%)
VAX_PERCENT_DISTRIBUTION	float	469	Percent of people getting vaccinated (%)
MEDIAN_HOUSEHOLD_INCOME	float	580	Median household income in dollars ($)
PERCENT_ATTAIN_BACHELORS	float	580	Percent of people who are 25-year-old and have bachelor's degrees or higher (%)
WHITE	float	570	Percent of people who identify as white (%)
BLACK	float	570	Percent of people who identify as black (%)
RACE_OTHER	float	570	Percent of people who identify as other races (%)
HISPANIC	float	590	Percent of people who identify as Hispanic (%)
AVG_TEMP	float	0	Average temperature (ºF)
PERCENT_VOTED_DEMOCRATIC	float	0	Percent of people who voted for demographic (%)
PERCENT_VOTED_REPUBLICAN	float	0	Percent of people who voted republican (%)
PERCENT_UNINSURED	float	0	Percent of people who do not have insurance (%)
PERCENT_PRIVATE_INSURANCE	float	0	Percent of people who have private insurance (%)
PERCENT_PUBLIC_INSURANCE	float	0	Percent of people who have public insurance (%)

C. Output: What are you trying to predict? Define the output clearly.

Our project aims to predict two outputs:

- The first output will be the predicted monthly search engine behavior for the next flu season, from July 2022 to April 2022, for each of the ten states. The leading search engine to be predicted is Google Trends with the most related search term: "flu shot". Therefore, for each interested state, our model would predict 10 data points for search engine behavior. The result would be 100 predicted data points over ten states. The predicted search engine behavior would be in float format to indicate the popularity of the search term over the given time.

- The second output will be the predicted change in the monthly vaccination rate for two flu seasons, from July 2021 to April 2023 and from July 2022 to April 2023, for each of the ten states. Each state has 20 predicted monthly data points in these periods. Therefore, a total of 200 predicted data points would be generated for all ten states. Each of the predicted changes in vaccination rate would be in float format, indicating the percent change in vaccination rate of a state compared to its previous month.
- If time allows, the project will deliver a digital dashboard displaying past and predicted future flu shot search queries and the change in vaccination rate given a time period. The digital dashboard will be helpful for anyone interested and make the project easier to access for further development in the future.

Part I. Google Trends Modeling

2. Predictive Models

A. What are the methods? Give a general introduction of the methods with references

ARIMA/SARIMA: ARIMA is an Autoregressive Integrated Moving Average algorithm for time series forecasting. ARIMA considers the past values and predicts future values based on that. However, ARIMA is limited, and does not support seasonal data or a time series with a repeating cycle. Predictable patterns that appear over a calendar year, could negatively affect the regression model (Hayes, 2021). SARIMA is a similar algorithm for forecasting, it is an extension of ARIMA, but more powerful and considers seasonality patterns. SARIMA stands for Seasonal Autoregressive Integrated Moving Average. In many time series data, seasonal effects come into play so there is a cyclical rise and fall or seasonality cycles in the data. SARIMA will take the seasonality cycles into consideration to predict future values (Yiu, 2021).

RNN: For our second method, we attempted to implement a Recurrent Neural Network to predict the result of variable 'FLU SHOT'. We take out the data from the state of Connecticut and create a time series. Then we feed the series to train the model. A recurrent neural network is a class of artificial neural networks where connections between nodes form a directed or undirected graph along a temporal sequence. This allows it to exhibit temporal dynamic behavior. We also use the parameter grid to try out different combinations of hyperparameters including different initializers, activation functions, optimizers, loss, batch sizes and epochs.

RNN LSTM: LSTM stands for Long Short-Term Memory network. An LSTM network is a type of recurrent neural network that is considered deep learning, and has been used in speech recognition and other forecasting problems based on its ability to learn order dependencies. They are different than traditional feed-forward neural networks because unlike regular neural networks, they are able to handle the "vanishing gradient problem". The vanishing gradient problem is the phenomenon where, as the algorithms travels backwards in the network, the training error signal exponentially decreases. The effect of this is that the layers closest to the input do not get trained. (Weberna's Blog) In essence, LSTMs are able to allow information in the model to persist and not be forgotten as the network is iterated through. Rather than a single neural network layer, LSTM has four: forget gate, input gate, cell state, and output gate.

B. Describe the methods with a pseudo code using the definitions in Section 1.

ARIMA/SARIMA:

- From our original dataset, three columns were extracted for this model, the "YEAR" column, the "MONTH" column, and the google query with the highest correlation to our vaccine distribution target variable, the "FLU SHOT" column.
- Join the "YEAR" and "MONTH" column, rename to "period"
- Make "period" the index of our new dataframe
- Look at the "FLU SHOT" plot over time visualizing the time series data
- Identify if the data is stationary or non-stationary

    - Run tests such as the Augmented Dickey Fuller (ADF) Test and Kwiatkowski–Phillips–Schmidt–Shin (KPSS) test for testing a null hypothesis that an observable time series is stationary around a deterministic trend.
    - If the data is non-stationary, make the time series stationary by differencing.
- Create autocorrelation plots
- Split data (train and test)
- Construct ARIMA and SARIMA models based on the data
- Forecast the next season of searching for the query "FLU SHOT"
- Plot forecast with 99% confidence intervals
- Use Evaluation Metrics to Measure Performance such as Root Mean Square Error (RMSE), visualization of plots with the 99% confidence interval, and R^2 value.
- Compare evaluation metrics to other models/algorithm

RNN

- Join the "YEAR" and "MONTH" column, rename to "YEAR-MONTH"
- Make the "YEAR-MONTH" as datetime object
- Turn the dataframe into a dart series
- Split the data into train and test
- Construct the RNN network
- Forecast the next season of searching for the query "FLU SHOT"
- Plot forecast
- Use Evaluation Metrics to Measure Performance such as Root Mean Square Error (RMSE), and R^2 value.
- Compare evaluation metrics to other models/algorithms

LSTM

- Read in our csv data and transform the data into a dataframe.
- use MinMaxScaler to scale the data from 0 to 1
- Split the data into testing and training sets
- Use the first 48 rows as training, and everything else as testing.
- Create a variable called "n_features" that is equal to 1, our FLU SHOT variable.
- Preprocess the testing and training data using TimeseriesGenerator, which outputs the input to our model.
- Call the model using the relu activation function, with 500 epochs and 100 for the dimensionality of the output space. The default ADAM (Adaptive movement Estimation Algorithm) is used, and the loss function is mean squared error.
- The model is trained on reshaped training data, which takes into account the number of features and inputs.

C. Justify the choice of the method.

ARIMA/SARIMA:

SARIMA is a widely used technique in time series with analysis that contains a seasonal component to predict future values based on historical data. SARIMA has some advantages, making it a popular machine learning algorithm to select for seasonal time series data. Further, we see seasonality cycles in our data.

Advantage:

- Model is easy to understand and interpret
- It can be easily used through mainstream statistical software such as SAS, SPSS, Python
- Can be used after the seasonal time series are stationary and have no missing values and takes seasonality cycles into consideration to predict future values.

Disadvantage:

- The longer the forecast, the harder it will be to predict accurately

RNN:

We try to use the method because we are doing time series analysis and our first algorithm is explicitly modeling with the seasonality in mind. The second algorithm is trying to take an alternative approach to see if the algorithm can try to figure out the seasonality through the iterative process.

LSTM:

Because LSTMs are designed to have "persistent memory" of data, they allow even more parameters to be learned from the data. LSTMs have been called "state-of-the-art models for forecasting" (Advanced Forecasting with Python, Korstanje). Therefore, we want to use this method to see if results differ from our other models.

3. Evaluations

A. What metrics do you use for evaluation?

a. Visualization: Plots with 99% confidence intervals. The actual values were plotted against the predicted values to visualize the models' accuracies. Additionally, a confidence interval helps us present the precision of our predictions. A 99% confidence interval means that 99% of the time the true population mean will be in that interval.

b. Root Mean Squared Error (RMSE) for the test dataset. RMSE is the standard deviation of the residuals (prediction errors). The smaller the RMSE, the better a given model is able to fit a dataset. RMSE is calculated using Equation 1:

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

where n is the total number of values in the test dataset. Y_i is the actual value at observation i, and $\widehat{Y_i}$ is the predicted value for observation i. RMSE is always positive, and the lower values are preferred. We calculate Root Mean Squared Error for each of our models to compare them.

c. R-squared (R^2) value. The R-squared value represents how close the regression line (predicted values plotted) is to the actual values. R-squared values range from 0 to 1. The closer the R-squared value is to 1 the better the model fits. R-squared (R^2) value is calculated using Equation 2:

$$R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \overline{y})^2}$$

Where SS_{RES} is the sum of squares of the residuals, and SS_{TOT} is the total sum of squares.

B. What is your ground truth?

Our ground truth is the Google Trends data using the search term "FLU SHOT" from April 2017 to April 2022. The ground truth is numerical data generated monthly for each of the ten states of interest. For example, for Connecticut in April 2017, the ground truth value is 2.0. This value indicates that the search term "FLU SHOT" in April 2017 is 2% compared to the highest time period of this search term which is 99% in October 2020 in the range from April 2017 to April 2022.

C. Discuss the performance and the limitations of the method.

ARIMA/SARIMA:

We performed the ADF and the KPSS test on our training data, consisting of 40 observations. Both tests resulted in our time series data being stationary. We first tried to use the ARIMA algorithm, however, it was not a good algorithm for the test set or the forecasting, therefore we continued with using SARIMA. When we used SARIMA for the evaluation of our test data set, it seemed like it closely followed the actual flu shot query numbers (Figure 1).

Therefore, we plotted our evaluation dataset with a 99% confidence interval, calculated the RMSE, and calculated and plotted the coefficient of determination (Figure 2). After, we were able to forecast the next year of flu shot query data using a 99% confidence interval. Below are our evaluation metrics for the SARIMA Model (Figure 3):

RMSE on SARIMA Test Data: 14.1

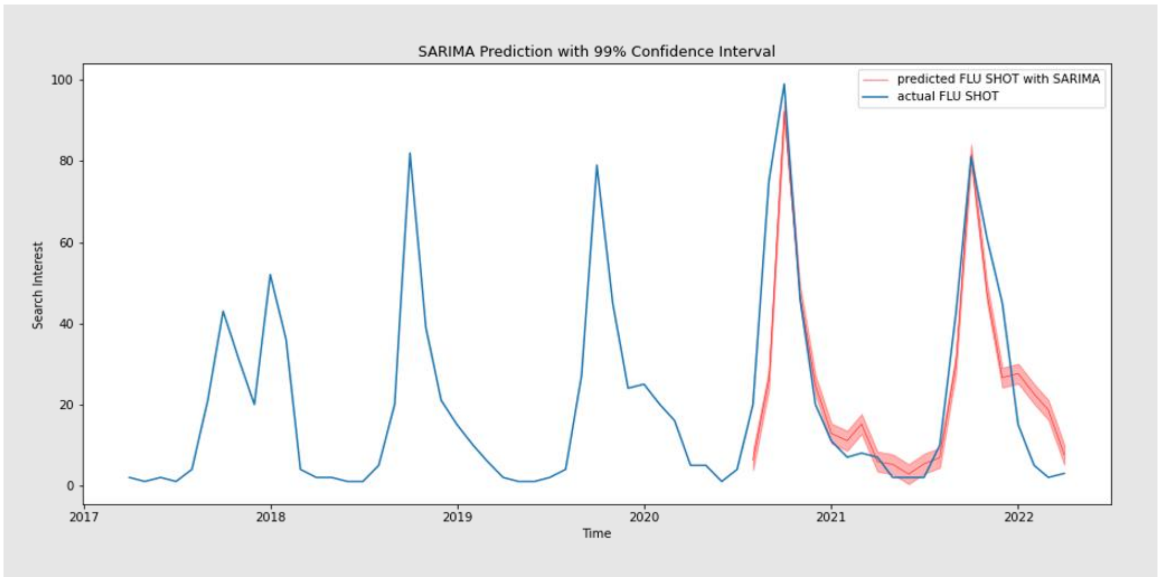

Figure 1. SARIMA prediction on test data

SARIMA Model R-squared (R^2): 0.767

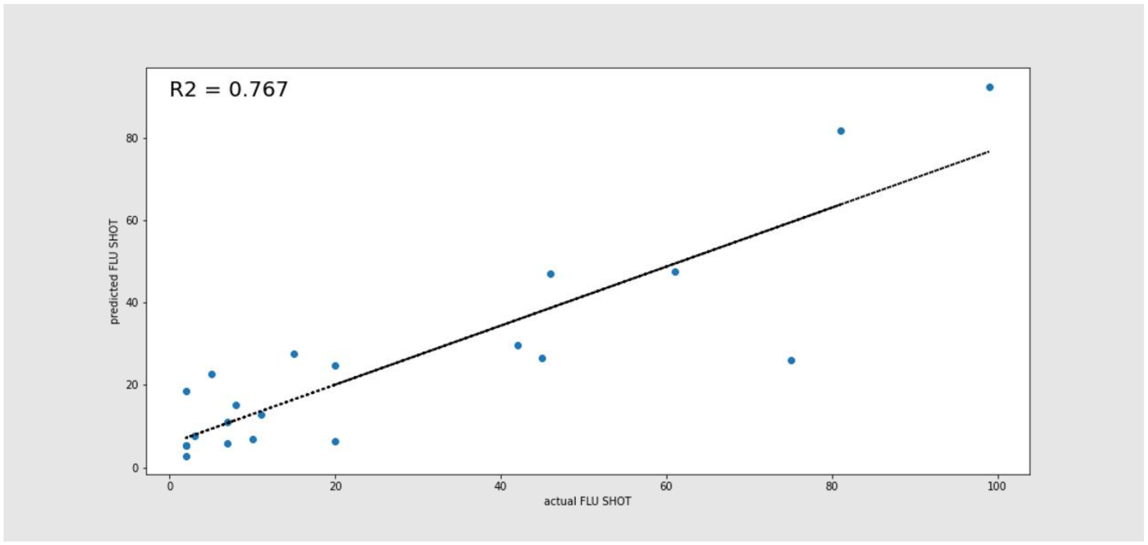

Figure 2. SARIMA prediction trendline and R^2

SARIMA Forecast:

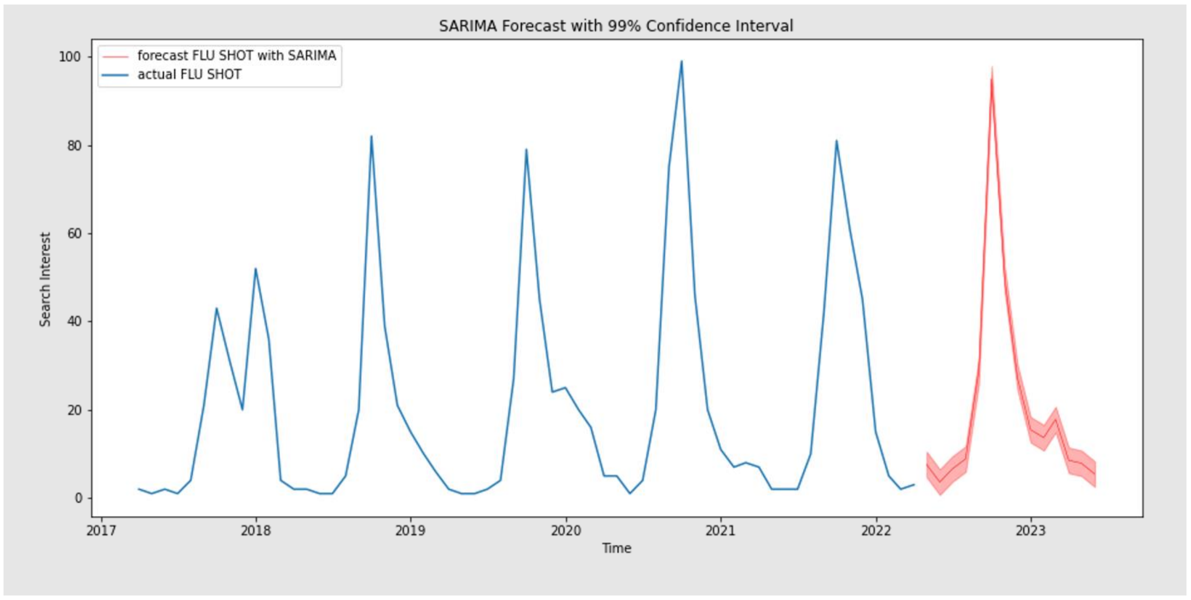

Figure 3. SARIMA forecast with 99% confidence interval

The RMSE of 14.1 indicates that on average our predicted values are only 14.1 (%) either lower or higher than the actual values. Figure 1 shows the predicted values were able to predict when the “FLU SHOT” search term is at peak which is at fall term, and when it dropped at the end of the years. The predicted values followed the cycle of the “FLU SHOT” search term. In Figure 3, SARIMA forecasted that there would be another peak for “FLU SHOT” search term around the fall season in 2022 and it would drop eventually toward the year of 2023. This forecasting is understandable since fall is usually the flu season when everyone would look for location or availability for a flu shot. Together with R^2 of 0.767 (Figure 2), SARIMA model performed well on predicting and forecasting the “FLU SHOT” search term on Google Trend. Moving forward, we decided to use this model as our baseline.

Limitations: SARIMA wouldn’t be accurate if we forecast for a longer period of time. SARIMA is based on the past data to predict the future data. In this project, our historical “FLU SHOT” search term is from 2017 to 2022. If we used our SARIMA model to predict the “FLU SHOT” search term for the next five years, 2023-2028, it would not be reliable anymore.

RNN

We have attempted to create an end-to-end pipeline to implement neural network with hyperparameter search build-in. The model captured the seasonality of the target variable and made accurate predictions, with RMSE of 18.07 and R squared of 0.5303.

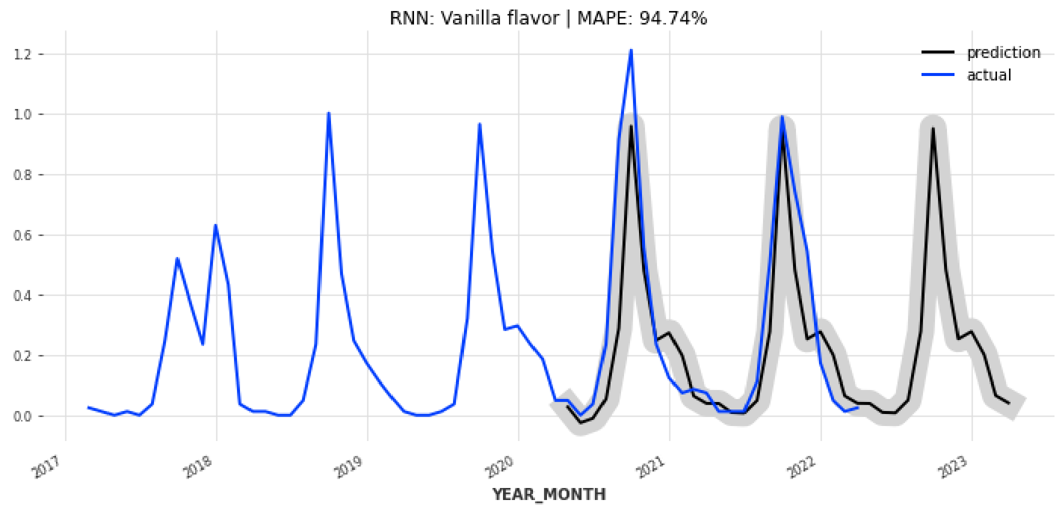

Figure 4. RNN prediction on test data

The advantage of the algorithm is that once the model is trained. It can be saved and deployed to solve similar problems. In addition, the model can pick up the seasonal trend without explicitly telling it. The limitation for the algorithm is that large amounts of data sometimes is required to achieve satisfactory result. Hyperparameter tuning is also obscure and hard to explain.

LSTM

The limitation of the model is that it can be prone to overfitting if there is not enough data. We are working with smaller datasets, so we need to be mindful of this limitation.

Hyperparameter tuning was conducted to obtain the best batch size, number of epochs, and number of neurons. Batch size determines the number of observations "shown" to the network before a weight is updated in the network. We tested the following batch sizes: 1, 2, 4. The number of epochs is the number of times the entire data frame has been passed through (and back) the algorithm. We tested the following number of epochs: 500, 1000, 3000, 4000, 5000. The number of neurons affects the learning capacity of the network. In general, more neurons will lead to more learning capacity, it can also lead to overfitting. We tested the following number of neurons: 1, 2, 3, 4, 5. Tuning revealed that the best parameters were a batch size of 4, with 1 neuron and 5000 epochs. Window size of 18 was used, and the relu activation function was chosen.

The best fit model had a rmse of 13.53 and an r-squared of 0.73. However, as you can see by the below graphs, the model “smoothed out” the fluctuations observed near the “valleys” of the graphs. For this reason, we chose the SARMIA method, since this model failed to capture that data variation.

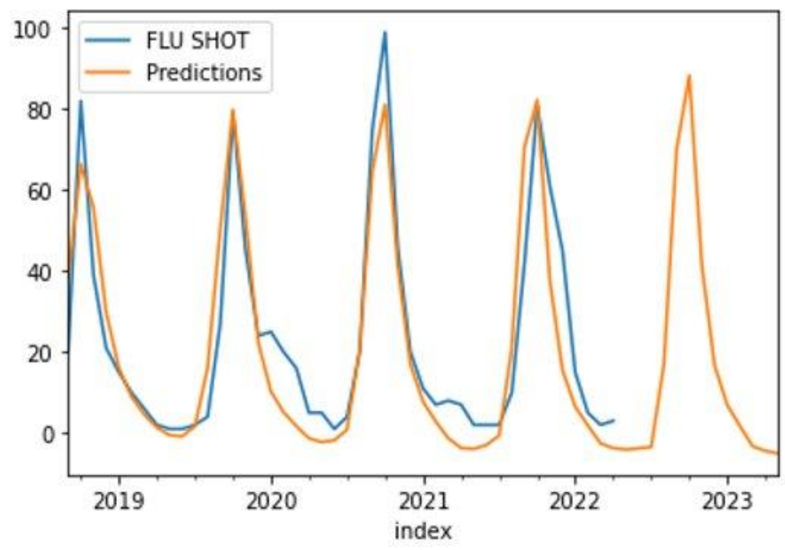

Figure 5. LSTM prediction on test data

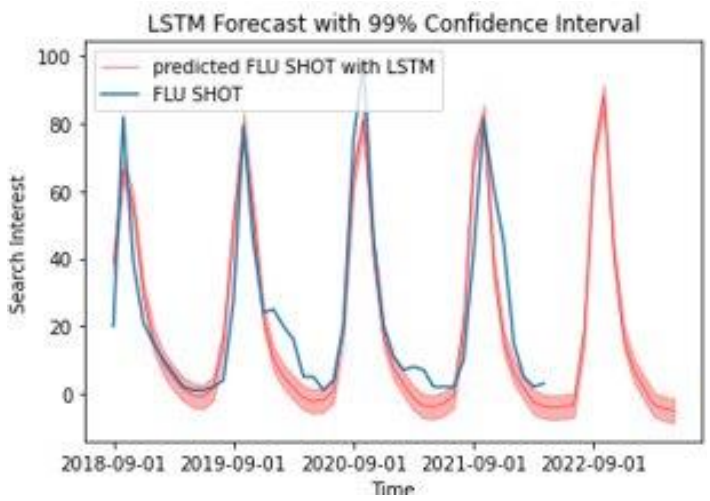

Figure 4. LSTM forecast with 99% confidence interval

The method of choice for algorithm 1 will be SARIMA, based on SARIMA's relatively low RMSE, high R-squared, and its ability to capture data fluctuations that other methods "smoothed" out. See the graph below for a comparison between all three methods.

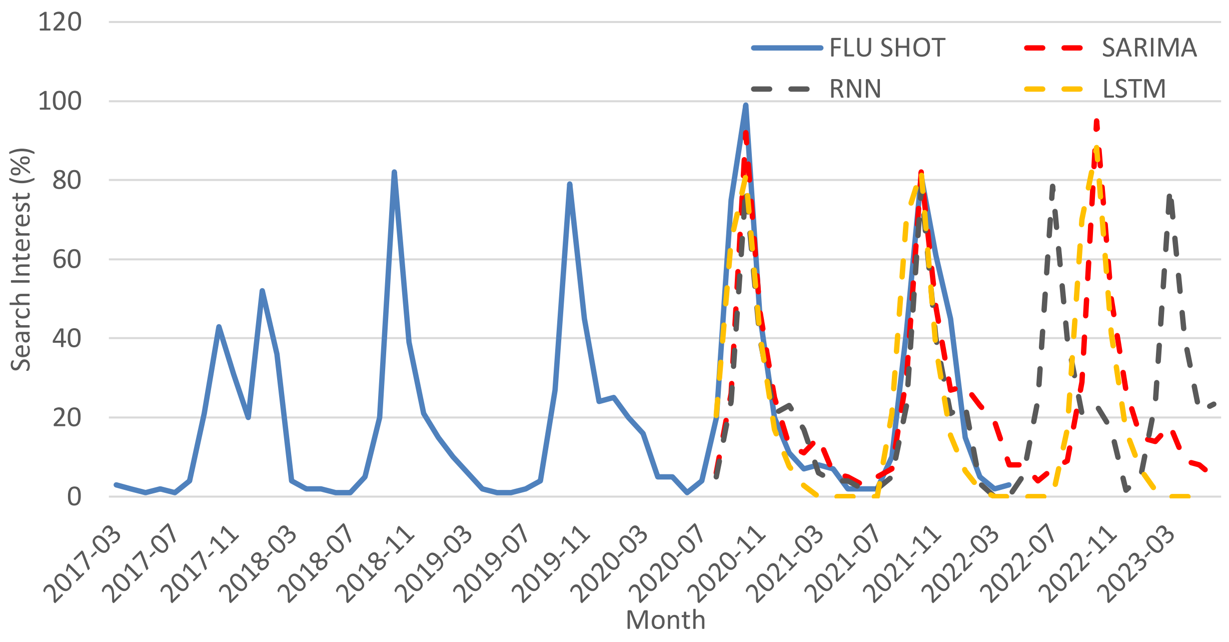

Figure 5. FLU SHOT with predictions of SARIMA, LSTM, and RNN

Part II: Vaccination Distribution

2. Predictive Models

A. What are the methods? Give a general introduction of the methods with references

LINEAR REGRESSION: A modeling approach that attempts to predict the value of a dependent variable based on an independent variable(s). Linear regression estimates the coefficients of the linear equation, involving one or more independent variables that best predict the value of the dependent variable. Linear regression fits a straight line or surface that minimizes the discrepancies between predicted and actual output values (*About Linear Regression | IBM*, n.d.).

DECISION TREE: A supervised modeling approach used to predict a target by learning decision rules from features (Li, 2019).

RANDOM FOREST: A random forest model combines the predictive power of multiple decision trees. It is a supervised machine learning algorithm that can be applied to both classification and regression models *(ibm.com)*.

GRADIENT BOOSTING: Gradient boosting is an ensemble machine learning method which uses boosting- a technique that builds from prior models, learns from their weights and loss functions to result in iteratively improved models (*towardsdatascience.com*).

B. Describe the methods with a pseudo code using the definitions in Section 1.

Data is split into training, validation, and prediction sets. Training contains rows of data 0-40. Validation contains rows of data 40-50, and prediction contains all rows of data after the 50th row.

The models are firstly trained with the training data using model.fit(x_train, y_train) function.

Then, the trained models are used to predict the validation set using model.predict(x_valid).

Finally, R^2 score for the validation is calculated for each model using sklearn.metrics package.

LINEAR REGRESSION:

LinearRegression model was conducted using sklearn.linear_model package with parameter fit_intercept was set to True.

DECISION TREE:

The decision tree model was conducted using sklearn.tree with a maximum depth parameter of 4.

RANDOM FOREST:

Hyperparameter tuning was conducted using sklearn.ensemble and RandomizedSearchCV packages. The following hyper parameters were considered:

- Number of trees in random forest model = [5,20,50,100]
- Number of features considered in each split = ['auto', 'sqrt']
- The maximum number of leaves in each tree = [every multiple of 10, starting at 10 through 140]
- The minimum sample number to split a node = [2, 6, 10]
- The minimum sample number that can be stored in a leaf node = [1, 3, 4]

The best parameters were found to be: n_estimators: 140, min_samples_split: 10, min_samples_leaf: 4, max_features: sqrt, max_depth: 90

GRADIENT BOOSTING:

Hyperparameter tuning was conducted using sklearn.ensemble and GridSearchCV packages. The following hyper parameters were considered:

- number of trees in the random forest:[100, 500,1000,2000]
- learning_rate:[.001,0.01,.1]
- The maximum number of leaves in each tree:[1,2,4]
- Fraction of samples used for each tree:[.5,.75,1]

The best parameters were found to be: 'learning_rate': 0.1, 'max_depth': 1, 'n_estimators': 100, 'subsample': 0.5, loss='squared_error'

C. Justify the choice of the method.

LINEAR REGRESSION: Linear regression will serve as a baseline model. Since this is a regression problem, we want to see how much of a difference there will be between a simple linear regression and more advanced models.

DECISION TREE: Decision Tree can be used as a regression method. A regression decision tree works by considering each feature in the training dataset. This method was considered because nonlinear relationships between parameters do not affect performance of the model.

RANDOM FOREST: Random Forest is an ensemble method for regression, which means it combines multiple predictions from multiple decision tree algorithms for a more powerful prediction. Random forest is less influenced by outliers- of which we have many in our most correlated variable to vaccination distribution: google trends flu shot. It is also very useful for feature selection because the model will not use unhelpful features in the model. We suspect some of our demographic variables may be superfluous, therefore Random Forest is a good model to utilize.

GRADIENT BOOSTING: The advantage of gradient boosting is that we can try out different loss functions during hyperparameter tuning in order to determine which is most optimal. Similar to random forest, it is very useful in feature selection *(Simic 2022)*.

3. Evaluations

A. What metrics do you use for evaluation?

R-squared (R^2) value. The R-squared value represents how close the regression line (predicted values plotted) is to the actual values. R-squared values range from 0 to 1. The closer the R-squared value is to 1 the better the model fits.

Root Mean Squared Error (RMSE). RMSE is the standard deviation of the residuals (prediction errors). The smaller the RMSE, the better a given model is able to fit a dataset.

B. What is your ground truth?

Our ground truth is the monthly CDC Vaccination Distribution **percent change** from April 2017 to April 2022. The ground truth is numerical data generated monthly for each of the ten states of interest and represents the **percent of the population that was vaccinated during that month**. For example, for Connecticut in April 2017, the ground truth value is 0.4, and the next month of May 2017 the value is 0.2. This means that in April 2017, an additional 0.4 percent of the population got vaccinated during that month. In May 2017, an additional 0.2 percent of the population got vaccinated during that month.

C. Discuss the performance and the limitations of the method.

Table 1. Evaluation Metrics for Four Models.

Model	**Average R^2 Train**	**Average R^2 Valid**	**Average RMSE Train (%)**	**Average RMSE Valid (%)**
Linear Regression	0.924	0.754	2.161	11.092
Decision Tree	0.995	0.612	0.149	14.544
Random Forest	0.984	0.838	0.499	7.022
Gradient Boosting	0.984	0.824	0.449	7.973

LINEAR REGRESSION: The limitation of linear regression is that it assumes that the independent variables (input variables) and dependent variable (predictor variables) are all linearly related- which may or may not be true. In fact, real world data very rarely has such simple relations. It also does not account for multicollinearity, which means when variables are not only correlated with the predictor variable, but with each other (Duke, 2022).

Performance: Across all ten states, all models had relatively high R squared values. Linear regression had the lowest average R squared value of all models. Across all states, linear regression had a moderate but comparatively low validation R squared to the other models, at an average of 0.75 across all models, and a RMSE value of 11.09.

DECISION TREE: When evaluating the performance of the training data, it was noticed that the decision tree model did not generalize the data well, therefore overfitting the training data. Because this model was overfitting the training data, we did not expect the performance on the validation set to be good, the R-squared validation value was 0.612, and the RMSE value for the validation set was the highest at 14.54.

RANDOM FOREST: Random forests can be relatively more difficult to interpret, since they involve multiple decision paths as well as multiple models.

Performance: Across all ten states, all models had relatively high R squared values. Random forest had a comparative validation performance to Gradient Boosting. Across all ten states, Random Forest had an average validation R squared of 0.838, and a RMSE of 7.02 for the validation set.

GRADIENT BOOSTING: One limitation of gradient boosting is that if we have very noisy data, it is possible that the model can begin to overfit *(Simic 2022)*.

Performance: Across all ten states, all models had relatively high R squared values. Gradient Boosting had a comparative validation performance to Random Forest, at an average of 0.824 across states. Gradient Boosting had also a comparative performance to Random Forest for the RMSE value at 7.97.

FINAL METHOD SELECTION: Due to their strikingly similar performance across gradient boosting and random forest in both our training and validation set evaluation, we decided to predict vaccine distribution using either gradient boosting or random forest: whichever model had the higher R-squared for the state in question. Figure 6 and 7, in the appendix, show a scatter plot of model performance for each state's training and validation dataset. Final vaccination distribution plot can be found on our Tableau dashboard: https://public.tableau.com/views/FluVaxPredictions/FluVaxPredictions?:language=en-US&:display_count=n&:origin=viz_share_link

Appendix

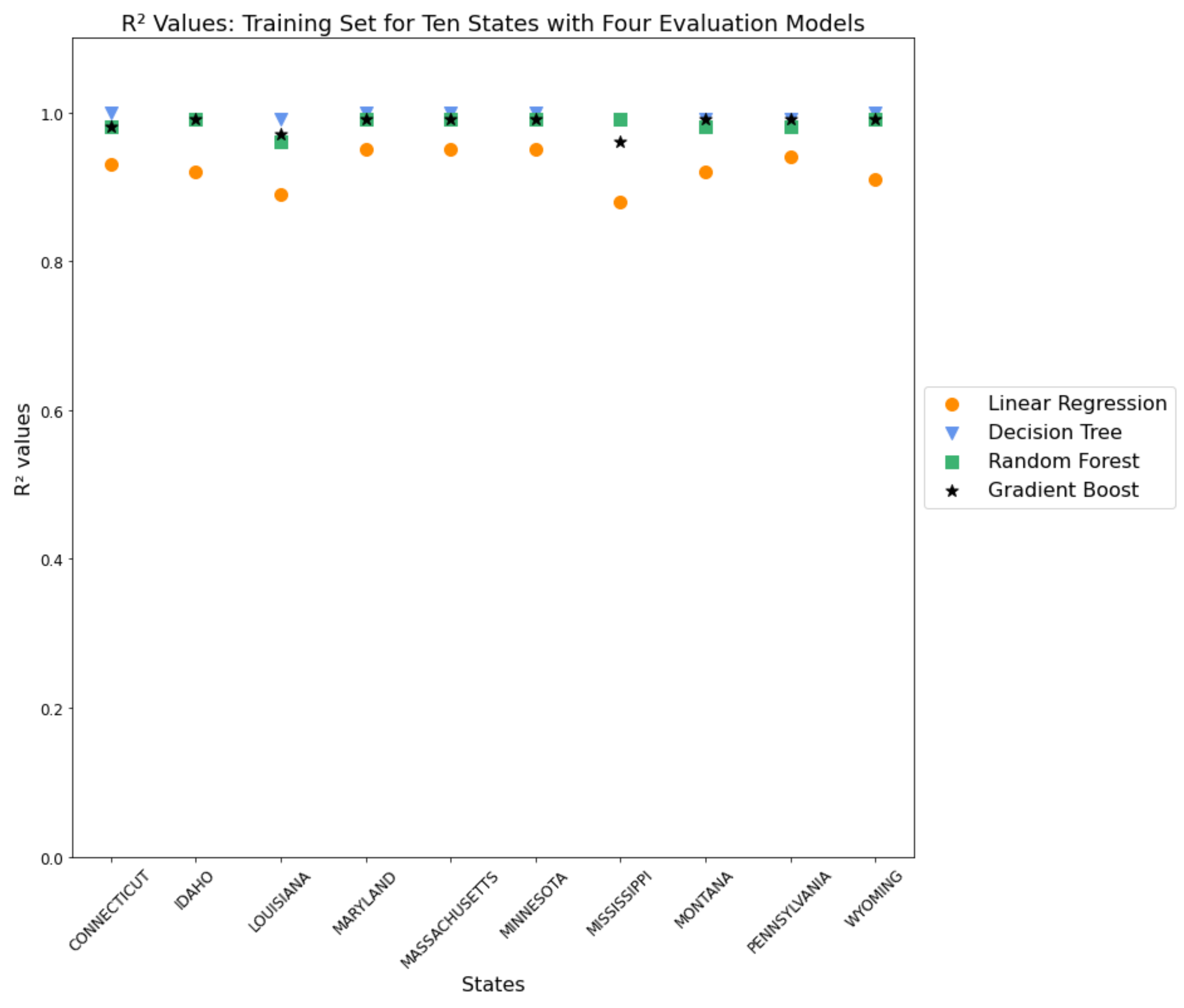

Figure 6. Evaluation of Four Models with the Training Dataset.

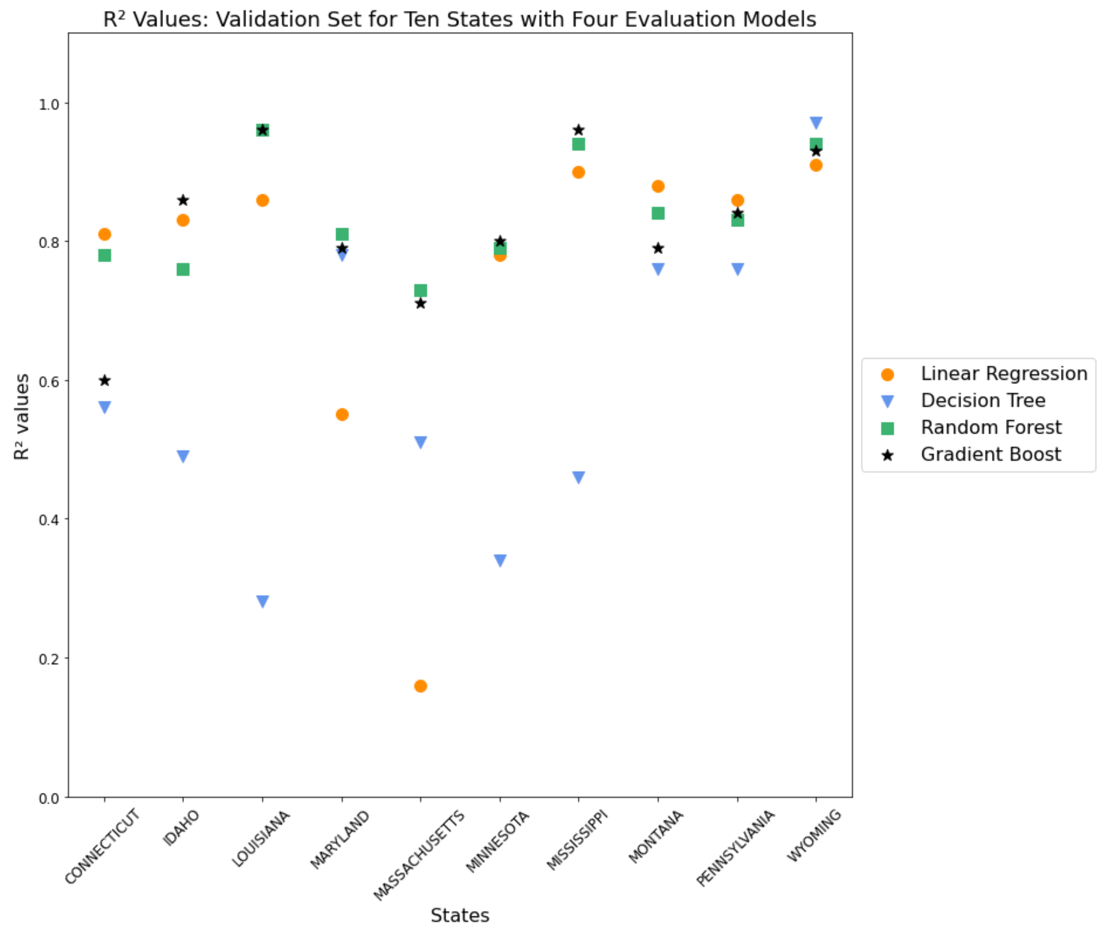

Figure 7. Evaluation of Four Models with the Validation Dataset.

Table of Contributions

The table below identifies contributors to various sections of this document.

	Section	**Writing**	**Editing**
1	**Predictive Modeling Problem Definition**	**Tien, Ashley, Jackie**	**Ashley, Tien , Jackie**
2	**Predictive Models**	**Tien, Ashley, Jackie, Andrew**	**Tien, Jackie, Ashley**
3	**Evaluations**	**Tien, Ashley, Jackie, Andrew**	**Ashley, Jackie, Tien**
4	**Appendix**	**Ashley**	

Sources:

About Linear Regression | IBM. (n.d.). Retrieved August 20, 2022, from https://www.ibm.com/topics/linear-regression

Hayes, A. (2021, October 12). *Autoregressive Integrated Moving Average (ARIMA)*. Investopedia. https://www.investopedia.com/terms/a/autoregressive-integrated-moving-average-arima.asp

IBM (2022) https://www.ibm.com/cloud/learn/random-forest#:~:text=Random%20forest%20is%20a%20commonly,both%20classification%20and%20regression%20problems.

Li, Lorraine. (2019, May 15). Classification and Regression Analysis with Decision Trees. *Towards Data Science*. Retrieved August 24, 2022, from https://towardsdatascience.com/https-medium-com-lorrli-classification-and-regression-analysis-with-decision-trees-c43cdbc58054

Simic, M. (2022, February 25). Gradient Bossting Trees vs. Random Forests. https://www.baeldung.com/cs/gradient-boosting-trees-vs-random-forests

Singh, H (2018m November 3). Understanding Gradient Boosting Machines. https://towardsdatascience.com/understanding-gradient-boosting-machines-9be756fe76ab

Stat at Duke. (Accessed August 23, 2022). Linear Regression. https://www2.stat.duke.edu/courses/Fall00/sta103/lecture_notes/multregr.pdf

Yiu, T. (2021, September 29). *Understanding SARIMA*. Medium. https://towardsdatascience.com/understanding-sarima-955fe217bc77

Weberna's blog. (Accessed July 29, 2022) *Why LSTMs Stop Your Gradients From Vanishing: A View from the Backwards Pass.* https://weberna.github.io/blog/2017/11/15/LSTM-Vanishing-Gradients.html#:~:text=If%20you%20don't%20already,input%20don't%20get%20trained.