H1N1 VACCINATION SHOT PREDICTION USING MACHINE LEARNING

By

NIKHIL KAUNDAL

TABLE OF CONTENTS

ABSTRACT

In the wake of the recurring threat posed by the COVID-19 virus, timely and accurate prediction of vaccination shot requirements is crucial for public health preparedness. This study proposes a machine learning-based approach to predict the demand for H1N1 vaccination shots. Leveraging historical vaccination data, demographic information, and epidemiological factors, a predictive model is developed to forecast the anticipated uptake of H1N1 vaccines within a specified population. The methodology integrates various machine learning algorithms, including decision trees, random forests, naive bayes and gradient boosting, to analyze past vaccination trends and identify key predictors influencing vaccination rates. Feature engineering techniques are employed to extract relevant features from the data source, enhancing the model's predictive performance. Validation of the predictive model is conducted using performance metrics such as accuracy, precision, recall, F1-score and receiver operating characteristic (ROC) analysis

The study demonstrate the effectiveness of the proposed approach in accurately forecasting H1N1 vaccination demand, thereby enabling healthcare authorities to optimize resource allocation, plan targeted vaccination campaigns, and mitigate the spread of the H1N1 virus. This research contributes to enhancing public health preparedness and response strategies for combating infectious disease outbreaks.

Keywords: H1N1 virus, Vaccination Prediction, Feature Engineering, Machine learning techniques, Machine learning algorithms, Random Forest, Support Vector Machine, Logistic Regression, K-Nearest Neighbours, Decision Tree, Naive Bayes and Gradient Boosting Classifier

CHAPTER-I

1. INTRODUCTION

1.1 The H1N1 Virus

In 1918, a deadly influenza pandemic caused by H1N1 influenza virus, also known as the Spanish flu, infected approximately 500 million people around the world and resulted in the deaths of 50 to 100 million people (3% to 5% of the world population) worldwide, distinguishing it as one of the deadliest pandemics in human history. In 2009, a new strain H1N1 swine flu quickly spread across the globe. A distinctive combination or integration of influenza genes was discovered in this novel H1N1 virus which was not identified prior in humans or animals.

This contagious novel virus had a very powerful impact on the whole world and spread across the world like a forest fire and as a result on June 11 2009 the World health Organization (WHO) declared that a pandemic of 2009 H1N1 flu or swine flu had begun . The effects of this novel H1N1 virus were more severe on people below the age of 65. There was significantly high paediatric mortality, and higher rate of hospitalizations for young adults and children, so the hospitals would soon get overwhelmed quickly as compared to non-pandemic influenza seasons. In August 2010, WHO declared the pandemic over. But the H1N1 flu strain from the pandemic became one of the strains that cause seasonal flu. Researchers estimate that in the first year, it was responsible for between 151,000 to 575,000 deaths globally.

A vaccine for the H1N1 flu virus became publicly available in October 2009. In late 2009 and early 2010, the United States conducted the National 2009 H1N1 Flu Survey. This phone survey asked respondents whether they had received the H1N1 and seasonal flu vaccines, in conjunction with questions about themselves. These additional questions covered their social, economic, and demographic background, opinions on risks of illness and vaccine effectiveness, and behaviours towards mitigating transmission. A better understanding of how these characteristics are associated with personal vaccination patterns can provide guidance for future public health efforts.

1.2 Objective of the Study

The primary objective of this study is to develop and compare various machine learning models for vaccination shot prediction.

The study aims to:

- Understand the factors influencing higher number of vaccinations.
- Evaluate the performance of different machine learning algorithms in predicting the trend of the vaccination.
- Provide actionable insights and recommendations to builders for reducing the consumption of energy so that energy can be conserved.

1.3 Scope of the Study

This study focuses on the application of machine learning algorithms to predict energy consumption while building and demolishing a building. The scope of the study includes:

- Data collection and preprocessing.
- Model development using various machine learning algorithms.
- Performance evaluation and comparison of the models.

1.4 Significance of the Study

The significance of this study lies in its potential to healthcare authorities to optimize efficient resource allocation, timely intervention, and effective disease control. By anticipating demand, public health agencies can optimize outreach efforts, enhance preparedness, and minimize the impact of outbreaks, ultimately safeguarding public health and promoting cost-effective strategies.

CHAPTER-II

2. LITERATURE REVIEW

2.1. Literature Review: Machine Learning Based Prediction of H1N1 and Seasonal Flu Vaccination

Reference: Srividya Inampudi “Machine Learning Based Prediction of H1N1 and Seasonal Flu Vaccination” In book: Advanced Computing (pp.139-150)

February 2021

DOI:10.1007/978-981-16-0401-0_11

1.Introduction and Background:

The H1N1 Flu that came into existence in 2009 in the U.S. and spread to the rest of the world had a great impact on the lives of people around the world. It was a life-threatening season to hundreds of people mainly below 65 years old which eventually made the World Health Organization (WHO) to declare it as the greatest pandemic in more than 40 years. The vaccines for H1N1 were first publicly available in the United States in October 2009, when the United States government began a vaccination campaign and people took vaccination based on certain factors.

2. Previous Research:

The paper references several studies that have applied machine learning techniques to study the H1N1 virus:

- Mabrouk "A chaotic study on pandemic and classical (H1N1) using EIIP sequence indicators, ": Described in their paper that the methods such as Moment invariants, correlation dimension, and largest Lyapunov exponent which were used to detect H1N1 and indicated the differences between the pandemic and classical influenza virus.
- Chrysostomou, In their paper the “Signal-processing-based bioinformatics approach for the identification of influenza A virus subtypes in Neuraminidase genes” majorly spoke about the Neuraminidase genes, Signal Processing, F-score, Support Vector Machines are the methods used for the identification of influenza virus

- Bao in their paper "Influenza-A Circulation in Vietnam through Data Analysis of Hemagglutinin Entries" provided NCBI influenza virus resources which provides many datasets (2001-2012) which is used for the analysis of influenza virus.

3. Methodology and Techniques:

The study proposed a system using machine learning techniques like Random Forest, Support Vector Machine and Logistic Regression for vaccination prediction.

Key Findings:

- Random Forest achieved a roc score of 0.8213.
- SVM achieved a roc score of 0.8397.
- Logistic Regression achieved a roc score of 0.8363.

4. Key Findings:

- It was observed that the younger population was more affected than the population aged above 65.
- The accuracy obtained with ANN for seasonal vaccine is 86.10%.Other machine learning algorithms have also yielded comparatively good results except logistic regression which has been the worst performing model with accuracy less than 70% in both H1N1 flu and seasonal flu vaccination prediction.

5. Conclusion

In this paper, prediction of H1N1 and seasonal flu vaccination are based on the data source given by the Nation H1N1 flu survey"2009"(NHFS) and center of disease control (CDC) and prediction of H1N1 vaccination is done best by the help of SVM model with RBF kernel with the help of hyperparameter tuning using GridSearchCV which yielded an accuracy of 83.97% and seasonal flu vaccination prediction is done best with Artificial neural networks which yielded an accuracy of 86.10%. This vaccine helps in protection from the H1N1 virus and seasonal flu. Awareness was created in all kinds of social media in order to give the required information to the people regarding the importance of the H1N1 vaccine and seasonal flu vaccine in 2009. In order to immunize the society and to provide a good safeguard environment for everybody, vaccinations were provided.

CHAPTER-III

METHODOLOGY

3.1 Data Collection and Preprocessing

3.1.1 Data Collection

The dataset utilized for this study is collected from the website of Centers for Disease Control and Prevention (National Public Health Agency of the United States) https://www.cdc.gov/nchs/nis/data_files_h1n1.htm. This dataset encompasses details taken by a phone survey which asked respondents whether they had received the H1N1 and seasonal flu vaccines, in conjunction with questions about themselves. These additional questions covered their social, economic, and demographic background, opinions on risks of illness and vaccine effectiveness, and behaviors towards mitigating transmission.

3.1.2 Data Preprocessing

The original dataset comprised 70,944 entries and 171 features. The following steps were executed to preprocess the dataset, ensuring it was well-suited for machine learning model implementation.

Data Loading:

The dataset was imported into a Pandas DataFrame, a widely used Python library for data manipulation and analysis.

Data Overview:

- The dataset consists of 70,944 entries and 171 features.
- The features include a mix of numerical and categorical variables.

Feature Selection:

As the data contains 171 columns but all are not needed for prediction , so the features related to their social, economic, and demographic background, opinions on risks of illness and vaccine effectiveness, and behaviors towards mitigating transmission were included.

After dropping the rest of the features ,39 features were left for model building.

Data Cleaning:

- **Checking Null Values:**
  Initial examination of the dataset revealed some columns having more than 20 % missing values.
- **Dropping Rows having Null Values:**
  Rows from the columns "CONCERN_DKNW_F', 'CONCERN_NONE_F', 'CONCERN_NOTV_F', 'CONCERN_REFD_F', 'CONCERN_SOME_F', 'CONCERN_VERY_F', 'KNOW_H1N1_ALOT_F', 'KNOW_H1N1_DKNW_F', 'KNOW_H1N1_LITL_F', 'KNOW_H1N1_NONE_F', 'KNOW_H1N1_REFD_F' " having missing values were removed.
- **Dropping Columns:**
  Columns "INSURE', 'Q95_INDSTR' , 'INC_CAT1" having High percentage of missing values were removed.
- **Imputing Missing Values:**
  The missing values in the reset of the columns were imputed using mode for categorical features and median for numerical features.

- The dataset after data cleaning has 56,656 entries and 36 features.

Feature Engineering:

New feature "target variable " was created from Vacc_H1N1_COUNT column where count of more than 1 was take as 1 and count of 0 was taken as 0.

The Features retained are:

BEHAVIORAL INDICATOR

- B_H1N1_ANTIV: TAKING ANTIVIRAL MEDICATIONS
- B_H1N1_AVOID: AVOID CLOSE CONTACT WITH OTHERS WITH FLU LIKE SYMPTOMS
- B_H1N1_FMASK: BOUGHT A FACE MASK
- B_H1N1_HANDS: WASHING HANDS
- B_H1N1_LARGE: REDUCED TIME AT LARGE GATHERINGS
- B_H1N1_RCONT: REDUCED CONTACT OUTSIDE THE HOME
- B_H1N1_TOUCH: AVOID TOUCHING EYES, NOSE, OR MOUTH

OPINION INDICATOR

- HQ23: EFFECTIVENESS OF H1N1 VACCINE
- HQ24: RISK OF GETTING SICK WITH H1N1 FLU WITHOUT VACCINE
- HQ24_B: WORRY ABOUT GETTING SICK FROM THE H1N1 VACCINE

CONCERN INDICATOR

- CONCERN_DKNW_F: H1N1 CONCERN LEVEL UNKNOWN
- CONCERN_NONE_F: NOT AT ALL CONCERNED ABOUT H1N1 FLU
- CONCERN_NOTV_F: NOT VERY CONCERNED ABOUT H1N1 FLU
- CONCERN_REFD_F:H1N1 CONCERN LEVEL REFUSED
- CONCERN_SOME_F: SOMEWHAT CONCERNED ABOUT H1N1 FLU
- CONCERN_VERY_F: VERY CONCERNED ABOUT H1N1 FLU

KNOWLEDGE INDICATOR

- KNOW_H1N1_ALOT_F: A LOT OF KNOWLEDGE ABOUT H1N1 FL
- KNOW_H1N1_DKNW_F: KNOWLEDGE LEVEL ABOUT H1N1 FLU UNKNOWN
- KNOW_H1N1_LITL_F: A LITTLE KNOWLEDGE ABOUT H1N1 FLU
- KNOW_H1N1_NONE_F: NO KNOWLEDGE ABOUT H1N1 FLU
- KNOW_H1N1_REFD_F: KNOWLEDGE LEVEL ABOUT H1N1 FLU REFUSED

VACC_H1N1_COUNT: NUMBER OF H1N1 FLU VACCINATIONS

OTHER INDICATOR

- DOCREC_H1N1_F: DOCTORS RECOMMENDATION FOR H1N1 VACCINE
- INSURE: HAS HEALTH INSURANCE COVERAGE
- HEALTH_WORKER_F: WORKS IN HEALTH CARE FIELD FLAG
- CHRONIC_MED_F: CHRONIC MEDICAL CONDITION FLAG
- PATIENT_CONTACT_F: DIRECT PATIENT CONTACT FLAG
- CLOSE_UNDER6MO_F: CLOSE CONTACT WITH CHILD UNDER 6 MONTHS FLAG
- AGEGRP: AGE GROUP
- EDUCATION_COMP: ADULT SELF-REPORTED EDUCATION LEVEL

- INC_CAT1: HOUSEHOLD INCOME CATEGORY
- RENT_OWN_R: IS HOME RENTED OR OWNED
- SEX_I: GENDER OF PERSON
- MARITAL: MARITAL STATUS
- RACE_I_R: RACE WITH MULTIRACE CATEGORY
- Q95: WORK STATUS
- Q95INDSTR: EMPLOYMENT INDUSTRY TYPE CODE
- MSA3_I: 3-CATEGORY MSA STATUS
- HHS_REGION: HHS SURVEILLANCE REGION NUMBER

Target Variable Description:

- The target variable H1N1_TAKEN represents whether a person have taken a vaccine or not.
- Vaccinated: Yes, No
- Distribution of the Target Variable:
- Churn:
- No: 44,736 entries (78.96%)
- Yes: 11,506 entries (21.03%)

3.2 Exploratory Data Analysis and Visualizations

To better understand the data and its distribution, various plots were created:

Count Plot:

- This plot was used to visualize the distribution of both the classes Vaccinated and Not Vaccinated across the dataset.

Bar Plots:

- A bar plot between H1N1 vaccination and Doctor recommendation give insights into how doctor recommendation influences H1N1 vaccination rates.

- This plot provide insights into how belief in the effectiveness of the H1N1 vaccine correlates with the percentage of people who took the vaccine, categorized by different levels of belief.
- This plot provide insights into how the perception of risk associated with H1N1 correlates with the percentage of people who took the vaccine, categorized by different levels of risk perception.

Correlation Map:

- The heatmap visually represents the correlation between different features in the DataFrame. Positive correlations are indicated by warmer colours, while negative correlations are indicated by cooler colours.

3.3 Model Development

3.3.1 Feature Scaling and Splitting

After preprocessing the dataset, it was split into features (X) and target (y). The features were scaled using MinMaxScaler and the data was split into training and testing sets with a ratio of 80:20.

3.3.2 Model Selection and Training

Several classification algorithms were evaluated for predicting customer churn:

1. Logistic Regression:

Logistic regression is a supervised machine learning algorithm used for classification tasks where the goal is to predict the probability that an instance belongs to a given class or not.
Logistic regression is used for binary classification where we use sigmoid function, that takes input as independent variables and produces a probability value between 0 and 1.

Logistic Function – Sigmoid Function

The sigmoid function is a mathematical function used to map the predicted values to probabilities.

It maps any real value into another value within a range of 0 and 1. The value of the logistic regression must be between 0 and 1, which cannot go beyond this limit, so it forms a curve like the “S” form.

The S-form curve is called the Sigmoid function or the logistic function.

In logistic regression, we use the concept of the threshold value, which defines the probability of either 0 or 1. Such as values above the threshold value tends to 1, and a value below the threshold values tends to 0.

Assumptions of Logistic Regression

The dependent variable must be categorical in nature.

The independent variable should not have multi-collinearity.

Logistic Regression Equation

The Logistic regression equation can be obtained from the Linear Regression equation. The mathematical steps to get Logistic Regression equations are given below:

- We know the equation of the straight line can be written as:

$$y= b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

- In Logistic Regression y can be between 0 and 1 only, so for this let's divide the above equation by (1-y):

$$\frac{y}{1-y}\text{; 0 for y= 0, and infinity for y=1}$$

- But we need range between -[infinity] to +[infinity], then take logarithm of the equation it will become:

$$log\left[\frac{y}{1-y}\right] = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

2. Decision Tree Classifier:

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.

The decisions or the test are performed on the basis of features of the given dataset.

It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.

It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.

The decision tree operates by analyzing the data set to predict its classification. It commences from the tree’s root node, where the algorithm views the value of the root attribute compared to the attribute of the record in the actual data set. Based on the comparison, it proceeds to follow the branch and move to the next node.

The algorithm repeats this action for every subsequent node by comparing its attribute values with those of the sub-nodes and continuing the process further. It repeats until it reaches the leaf node of the tree.

3. Extra Tree Classifier:

The extra trees algorithm, like the random forests algorithm, creates many decision trees, but the sampling for each tree is random, without replacement. This creates a dataset for each tree with unique samples.

Extremely Randomized Trees Classifier (Extra Trees Classifier) is a type of ensemble learning technique which aggregates the results of multiple de-correlated decision trees collected in a “forest” to output its classification result.

In concept, it is very similar to a Random Forest Classifier and only differs from it in the manner of construction of the decision trees in the forest.

Each Decision Tree in the Extra Trees Forest is constructed from the original training sample. Then, at each test node, Each tree is provided with a random sample of k features from the feature-set from which each decision tree must select the best feature to split the data based on some mathematical criteria (typically the Gini Index). This random sample of features leads to the creation of multiple de-correlated decision trees.

4. Random Forest Classifier:

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

Assumptions for Random Forest

Since the random forest combines multiple trees to predict the class of the dataset, it is possible that some decision trees may predict the correct output, while others may not. But together, all the trees predict the correct output. Therefore, below are two assumptions for a better Random Forest classifier:

- There should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result.
- The predictions from each tree must have very low correlations.

Why use Random Forest?

- It takes less training time as compared to other algorithms.
- It predicts output with high accuracy, even for the large dataset it runs efficiently.
- It can also maintain accuracy when a large proportion of data is missing.

5. Naive Bayes

The Naive Bayes algorithm is a supervised machine learning algorithm. It uses the Bayes Theorem to predict the posterior probability of any event based on the events that have already occurred. Naive Bayes is used to perform classification and assumes that all the events are independent. The Bayes theorem is used to calculate the conditional probability of an event, given that another event has already occurred.

Bayes' Theorem

Bayes' Theorem finds the probability of an event occurring given the probability of another event that has already occurred. Bayes' theorem is stated mathematically as the following equation:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

where A and B are events and P(B) ≠ 0.

Basically, we are trying to find probability of event A, given the event B is true. Event B is also termed as evidence.

P(A) is the priori of A (the prior probability, i.e. Probability of event before evidence is seen). The evidence is an attribute value of an unknown instance (here, it is event B).

P(B) is Marginal Probability: Probability of Evidence.

P(A|B) is a posteriori probability of B, i.e. probability of event after evidence is seen.

P(B|A) is Likelihood probability i.e the likelihood that a hypothesis will come true based on the evidence.

Types of Naive Bayes Model:

These are the types of Naive Bayes Model, which are given below:

- Gaussian: The Gaussian model assumes that features follow a normal distribution. This means if predictors take continuous values instead of discrete, then the model assumes that these values are sampled from the Gaussian distribution.
- Bernoulli: The Bernoulli classifier works similar to the Multinomial classifier, but the predictor variables are the independent Booleans variables. Such as if a particular word is present or not in a document. This model is also famous for document classification tasks.

6. KNN Classifier:

K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.

K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.

K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.

K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.

It is widely disposable in real-life scenarios since it is non-parametric, meaning it does not make any underlying assumptions about the distribution of data.

It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.

The K-NN algorithm works by finding the K nearest neighbors to a given data point based on a distance metric, such as Euclidean distance. The class or value of the data point is then determined by the majority vote or average of the K neighbors. This approach allows the algorithm to adapt to different patterns and make predictions based on the local structure of the data.

How does K-NN work?

The K-NN working can be explained on the basis of the below algorithm:

- Step-1: Select the number K of the neighbors
- Step-2: Calculate the Euclidean distance of K number of neighbors
- Step-3: Take the K nearest neighbors as per the calculated Euclidean distance.
- Step-4: Among these k neighbors, count the number of the data points in each category.
- Step-5: Assign the new data points to that category for which the number of the neighbor is maximum.
- Step-6: Our model is ready.

8. Gradient Boosting Classifier:

Gradient Boosting is a powerful boosting algorithm that combines several weak learners into strong learners, in which each new model is trained to minimize the loss function such as mean squared error or cross-entropy of the previous model using gradient descent.

In each iteration, the algorithm computes the gradient of the loss function with respect to the predictions of the current ensemble and then trains a new weak model to minimize this gradient. The predictions of the new model are then added to the ensemble, and the process is repeated until a stopping criterion is met.

GBT builds decision trees sequentially.

Each new tree in the ensemble focuses on reducing the errors made by the previous ones.

The algorithm fits each new tree on the residual errors (the difference between the predicted and actual values) of the previous ensemble.

This sequential nature allows GBT to learn complex relationships in the data but makes it more prone to overfitting, especially if not properly regularized.

9. XGBoost Classifier:

XGBoost is an optimized distributed gradient boosting library designed for efficient and scalable training of machine learning models. It is an ensemble learning method that combines the predictions of multiple weak models to produce a stronger prediction. XGBoost stands for "Extreme Gradient Boosting" and it has become one of the most popular and widely used machine learning algorithms due to its ability to handle large datasets and its ability to achieve state-of-the-art performance in many machine learning tasks such as classification and regression.

One of the key features of XGBoost is its efficient handling of missing values, which allows it to handle real-world data with missing values without requiring significant pre-processing. Additionally, XGBoost has built-in support for parallel processing, making it possible to train models on large datasets in a reasonable amount of time.

The optimization method (gradient) minimizes a cost function by repeatedly changing the model's parameters in response to the gradients of the errors. The algorithm also presents the idea of "gradient boosting with decision trees," in which the objective function is reduced by calculating the importance of each decision tree that is added to the ensemble in turn. By adding a regularization term and utilizing a more advanced optimization algorithm, XGBoost goes one step further and improves accuracy and efficiency.

10. CatBoost Classifier:

CatBoost or Categorical Boosting is an open-source boosting library developed by Yandex. It is designed for use on problems like regression and classification having a very large number of independent features.

Catboost is a variant of gradient boosting that can handle both categorical and numerical features. It does not require any feature encodings techniques like One-Hot Encoder or Label Encoder to convert categorical features into numerical features. It also uses an algorithm called symmetric weighted quantile sketch(SWQS) which automatically handles the missing values in the dataset to reduce overfitting and improve the overall performance of the dataset.

How Catboost Works:

Catboost works on gradient boosting algorithms in which decision trees are constructed iteratively on each iteration and each tree improves the results of the previous trees leading to better results. The difference between catboost and other gradient boosting algorithms is that it handles the categorical features, performs cross-validation, regularization to avoid overfitting, etc, on its own which gives catboost an edge over other algorithms as no preprocessing is required.

Summary

Each of these classification algorithms offers unique advantages and characteristics that make them suitable for predicting H1N1 vaccination in this project. By employing a combination of these models, we aim to identify the most effective and robust model for accurately predicting H1N1 vaccination.

3.4 Hyperparameter Tuning

In the process of building predictive models for customer churn, hyperparameter tuning was an essential step to enhance the performance and accuracy of the models. Hyperparameters are parameters that are not learned from the data but are set before training the model. Tuning these hyperparameters is crucial to optimize the model's performance and generalization ability.

Logistic Regression:

For the Logistic Regression model, hyperparameters like the regularization strength (C), class weight and solver were optimized using GridSearchCV. No change in the training and testing scores was observed.

Decision Tree Classifier:

In the Decision Tree model, hyperparameters such as the max_depth, min_samples_split, min_samples_leaf and class weight were tuned. GridSearchCV was employed to search through various combinations of these hyperparameters to identify the optimal set that yields the highest accuracy. Training scores decreased and testing scores increased as overfitting was reduced.

Extra Tree Classifier:

In the Extra Tree model, hyperparameters such as the max_depth, min_samples_split, and min_samples_leaf were tuned. GridSearchCV was employed to search through various combinations of these hyperparameters to identify the optimal set that yields the highest accuracy. Training scores decreased and testing scores increased as overfitting was reduced.

Random Forest:

In the Random Forest model, hyperparameters such as the max_depth, min_samples_split, and min_samples_leaf were tuned. GridSearchCV was employed to search through various combinations of these hyperparameters to identify the optimal set that yields the highest accuracy . Training scores decreased and testing scores increased as overfitting was reduced.

Bernoulli Naive Bayes Classifier:

In the Bernoulli Naive Bayes Classifier model, hyperparameters such as the alpha and fit_prior were tuned. No change in the training and testing scores was observed.

Gaussian Naive Bayes:

Bernoulli Naive Bayes Classifier model does not support hyperparameter tunning.

K-Nearest Neighbors (KNN):

In the KNN model, the number of neighbors (n_neighbors) , distance metric and the weight function (weights) were the hyperparameters tuned. Training scores decreased and testing scores increased as overfitting was reduced.

Gradient Boosting Classifier:

For the Gradient Boosting Classifier learning_rate and max_depth was tunned. There was a decrease in both training and testing scores.

XGBoost Classifier:

For the XGBoost Classifier learning_rate and max_depth was tunned. There was a increase in both training and testing scores.

Cat Boost Classifier:

For the Cat Boost Classifier learning_rate and depth was tunned. There was a decrease in both training and testing scores.

Summary:

Hyperparameter tuning is a critical step in the machine learning pipeline to optimize the model's performance. By employing GridSearchCV, we were able to systematically search through the hyperparameter space and identify the optimal set of hyperparameters for each model. This optimization process enhances the predictive accuracy of the models and ensures better generalization on unseen data.

CHAPTER-IV

RESULTS AND DISCUSSION

4.1 EXPERIMENTAL RESULTS

The following section presents the detailed results obtained from implementing various machine learning algorithms to predict H1N1 vaccination based on the dataset.

1. Logistic Regression:

Before Hyperparameter Tunning

The Logistic Regression model achieved a test accuracy of 79.62%.

The ROC-AUC for this model is 0.71.

After Hyperparameter Tunning

The Logistic Regression model achieved a test accuracy of 79.62%.

The ROC-AUC for this model is 0.71.

2. Decision Tree:

Before Hyperparameter Tunning

The Decision Tree model achieved a test accuracy of 72.53%.

The ROC-AUC for this model is 0.60.

After Hyperparameter Tunning

The Decision Tree model achieved a test accuracy of 81.08%.

The ROC-AUC for this model is 0.76.

3. Extra tree classifier:

Before Hyperparameter Tunning

The Extra Tree model achieved a test accuracy of 72.14%.

The ROC-AUC for this model is 0.58.

After Hyperparameter Tunning

The Extra Tree model achieved a test accuracy of 79.1%.

The ROC-AUC for this model is 0.69.

4. Random Forest classifier:

Before Hyperparameter Tunning

The Random Forest model achieved a test accuracy of 81.21%.

The ROC-AUC for this model is 0.77.

After Hyperparameter Tunning

The Random Forest model achieved a test accuracy of 81.62%.

The ROC-AUC for this model is 0.79.

5. Bernoulli Naive Bayes:

Before Hyperparameter Tunning

The Bernoulli Naive Bayes model yielded a test accuracy of 77.96%.

The ROC-AUC for this model is 0.72.

After Hyperparameter Tunning

The Bernoulli Naive Bayes model yielded a test accuracy of 77.96%.

The ROC-AUC for this model is 0.72.

6. Gaussian Naive Bayes:

The Gaussian Naive Bayes model yielded a test accuracy of 71.35%.

The ROC-AUC for this model is 0.68.

7. KNN classifier:

Before Hyperparameter Tunning

The KNN model achieved a test accuracy of 76.72%.

The ROC-AUC for this model is 0.62.

After Hyperparameter Tunning

The KNN model achieved a test accuracy of 78.52%.

The ROC-AUC for this model is 0.63.

8. Gradient boosting classifier:

Before Hyperparameter Tunning

The Gradient Boosting model achieved a test accuracy of 81.79%.

The ROC-AUC for this model is 0.80.

After Hyperparameter Tunning

The Gradient Boosting model achieved a test accuracy of 81.72%.

The ROC-AUC for this model is 0.80.

9. XG Boost classifier:

Before Hyperparameter Tunning

The XG Boost model achieved a test accuracy of 81.59%.

The ROC-AUC for this model is 0.79.

After Hyperparameter Tunning

The XG Boost model achieved a test accuracy of 81.80%.

The ROC-AUC for this model is 0.80.

10. Cat Boost classifier:

Before Hyperparameter Tunning

The Cat Boost model achieved a test accuracy of 82.15%.

The ROC-AUC for this model is 0.80.

After Hyperparameter Tunning

The Cat Boost model achieved a test accuracy of 81.81%.

The ROC-AUC for this model is 0.80.

CHAPTER-V

CONCLUSION AND FUTURE SCOPE

5.1 CONCLUSION

In this study, prediction of H1N1 vaccination is based on the data source given by the National 2009 H1N1 Flu Survey (NHFS) and Centers for Disease Control and Prevention (CDC). Through experimental results, it was observed that all boosting algorithms—the CatBoost algorithm, the Gradient Boosting algorithm, and the XGBoost algorithm—provided the best ROC-AUC score of 0.80. However, the CatBoost model achieved a slightly higher test accuracy of 82.15%.

5.2 FUTURE SCOPE

In future work, the following aspects can be explored to enhance the predictive accuracy and robustness of the churn prediction models:

Feature Engineering: Incorporating more relevant features or creating new features from the existing ones to improve the model's predictive power.

Advanced Machine Learning Techniques: Exploring more sophisticated machine learning algorithms or ensemble methods to further boost the model's performance.

Integration of Real-Time Data: Incorporating real-time epidemiological data, social media signals, and healthcare utilization patterns can enhance the accuracy and timeliness of vaccination predictions.

By addressing these aspects, it is anticipated that the predictive accuracy and effectiveness of the vaccination prediction models can be further improved, leading to better epidemic preparedness, optimizing vaccination strategies, and ultimately mitigate the impact of H1N1 outbreaks on public health.