Humana-Mays Healthcare Analytics Competition

Members Hesitant to Covid Vaccine

Capstone Project In Analytics

Instructor: Amin Gharipour

Authors
Niharika Pillanagoyala
Sindhu Vegiraju

Table of Contents

1. Executive Summary

This study focused on helping Humana in increasing the COVID-19 vaccination rates among its members. It has been a continuous priority both for members and the larger population health and safety considerations. Humana is particularly focused on providing vaccination opportunities for the most vulnerable and underserved populations. Existing disparities in health equity have become very visible during the vaccine response to COVID-19.

Humana and many others have reduced real barriers to gaining access for those who want the vaccine. In addition, a portion of the members, like in the general population, are hesitant or resistant to get the vaccine due to a combination of factors that include, misinformation, and lack of trust in the vaccine. It is believed that these members will require personalized outreaches involving clinical conversations to build trust in the vaccine. However, member-level vaccination status data is incomplete due to a lack of consistent and timely data capture at the vaccination sites.

Our goal was to develop a model to predict a model that identifies members who are most likely to be hesitant to the covid vaccine so that Humana can design targeted outreaches for these members, prioritized to reach the most vulnerable and underserved populations to receive health solutions.

More importantly, we wanted to generate actionable insights and key indicators aligned to Humana's business needs.

For this study, firstly, we identified the target variable i.e., covid_vaccination (1 if the Member has taken the vaccine, 0 otherwise) and we noticed the classes were imbalanced. Secondly, we did some data preprocessing to drop highly correlated columns and to deal with NULL values to better capture the underlying trends in the data. After data processing, we tested different model architectures on our data to find the optimal model.

We obtained the end result of our best performing model – XGBoost and Decision Tree. We trained on XGB classifier and DecisionTree classifier using some hyperparameters and got a Receiver Operating Characteristic (ROC) Area Under Curve (AUC) of 1 and

an accuracy of 100%. We arrived at these models by maximising the ROC-AUC and minimising the False Positive Rate.

2. Data Preparation

2.1. Data Understanding

To perform our analysis, we were given a training dataset of 974842 observations and 368 variables and a Holdout dataset of 525,158 rows and 367 columns. The provided data was rolled-up at the member level with the information provided on 8 different types of events including Demographic, medical claims, pharmacy claims, lab claims, credit data, CMS features, clinical condition related features, and other features, with each event having its own set of attributes, for a total of 368 attributes. Following is one of the data nuances that we observed during processing the data:

- The data was highly imbalanced with the target variable in the ratio of 8,05,389 for class 0, i.e., people who had not taken covid vaccine; and 1,69,453 for class 1, i.e., people who got the covid vaccine.

Our objective is to identify the people who are hesitant to take vaccines and those who are willing to take vaccines based on the 367 variables. Though the variables affecting the target feel logically irrelevant, we will understand the relationship of each variable internally and consider only those variables which have a high correlation to the target.

2.2. Target Modelling

Our aim is to identify the Humana Members who are most likely hesitating to get covid vaccinated and to provide recommendations & potential solutions to drive vaccination among the sub-segments of hesitant members based on the insights derived from the data and overcome this barrier to accessing care and achieving their best health.

It is likely that non vaccinated members are not homogeneous and may have different reasons for not getting the vaccination (e.g. pre-existing conditions, access, misinformation, lack of trust in the vaccine). As such, there are perhaps different solutions for different segments of members.

Any member who is hesitant to get vaccinated would come up with a positive outcome.

2.3 Feature Engineering

We have divided the dataset into training and test data. Train dataset consists of all the variables of the original dataset except ‘covid_vaccination’ and the Test dataset consists of only one variable i.e., ‘covid_vaccination’.

We explored and extracted those feature columns where the missing data was greater than 70 percent. Columns that have greater missing data do not influence the target features of ‘covid_vaccination. Hence, we removed the following feature column:

● ‘lang_spoken_cd’

Next, we segregated the features as numerical and categorical features. We handled each numerical and categorical feature in a different way.

For the categorical features, we imputed the values of the missing features ‘Na’ with the mode value.

We selected all the categorical variables that contain object data type and considered cardinality for each categorical variable.

This was followed by a label encoding the categorical features which had ordinality. We have divided all the categorical variables into four lists and performed Label Encoding and then combined all the variables from all the lists.

For the numerical features, we performed Scaling using Minmax scaler. For each feature, each value is subtracted by the minimum value of the respective feature and then divided by the range of original maximum and minimum of the same feature. It has a default range between [0,1].

Here, lastly Label Encoded the test set which has only a single variable ‘covid_vaccination’.

Finally, concatenated the Train dataset and Test dataset into a dataset called ‘data’

Now, consider taking all the records from the minority group where the members are covid vaccinated and create a minority_sample dataset.

Perform undersampling on the majority group and use the low-frequency group count to randomly sample from high-frequency(majority) group and create a majority_sample dataset.

Finally concatenate minority_sample and majority_sample datasets into a single dataset called ‘merged_sample’.+Now the ‘merged_sample’ dataset is converted to a ‘preprocessed_data.csv’ file

All the above steps are repeated for the Holdout dataset and created a csv file called ‘preprocessed_holdout.csv’

3. Modelling

We tried the below models on our datasets.

To model our dataset, we trained all our models on a 70-30 train-test split. To validate our model we used the accuracy and ROC-AOC metric since our problem is a binary classification problem.

3.1. XGBOOST

XGBOOST is a decision-tree based ensemble Machine Learning algorithm that uses a gradient boosting framework.

XGBoost are both ensemble tree methods that apply the principle of boosting weak learners using the gradient descent architecture. However, XGBoost improves upon the base GBM framework through systems optimization and algorithmic enhancements.

Suppress Warnings

```
import warnings
warnings.filterwarnings('ignore')
```

Load Packages

```
# Importing desired libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# visualization
from matplotlib.pyplot import xticks
%matplotlib inline
```

Load Data

```
merged_sample = pd.read_csv('../preprocessed_data.csv')
merged_sample.head(5)
merged_sample_copy = merged_sample.copy()
```

Split into training and test datasets

In order to evaluate the performance we divide the data into training and test sets

```
train = merged_sample.drop(columns=['covid_vaccination'])
test = merged_sample_copy[['covid_vaccination']]
train.head()
```

We used the 70:30 ratio split for this dataset. Splitting the data into training and the test data

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(train, test,
test_size=0.3, random_state=0)
y_train.value_counts()
```

```
covid_vaccination
0                   118838
1                   118396
dtype: int64
```

```
y_test['covid_vaccination'].value_counts()
```

```
1    51057
0    50615
Name: covid_vaccination, dtype: int64
```

Instantiate the XGBClassifier() model and then fit the model on the training data.

```
#Import XGBOOST model
from xgboost import XGBClassifier

#Create a Gaussian Classifier
clf = XGBClassifier(random_state=42)

#Train the model using the training sets y_pred=clf.predict(X_test)
clf.fit(x_train,y_train)
```

Generate predictions

We can make predictions using the trained model on the test data

```
# prediction on test set
preds=clf.predict(x_test)
```

By default, the predictions made by XGBoost are probabilities. Because this is a binary classification problem, each prediction is the probability of the input pattern belonging to the first class.

```
import numpy
print(numpy.unique(preds))
preds
print(numpy.count_nonzero(preds == 1))
print(numpy.count_nonzero(preds == 0))
print(numpy.size)
y_test['covid_vaccination'].value_counts()
```

Evaluate model performance

```
from sklearn.metrics import
classification_report,confusion_matrix,accuracy_score
def elavutaionmetrix(x_train,y_train,y_test, preds):
    print(classification_report(y_test,preds))
    print("train accuracy:",clf.score(x_train,y_train))
    print("Test accuracy:",accuracy_score(y_test, preds))
```

Receiver Operating characteristic Curve(ROC) curves summarize the trade-off between the true positive rate and the false positive rate for a predictive model using different probability thresholds. ROC curves are appropriate when the observations are balanced between each class.
Precision-Recall curves summarize the trade-off between true positive rate and positive predictive value for a predictive model using different probability thresholds.
F-1 Score calculates the harmonic mean of the precision and recall.

```
precision    recall  f1-score   support

           0       1.00      1.00      1.00     50615
           1       1.00      1.00      1.00     51057
```

```
    accuracy                           1.00    101672
   macro avg       1.00      1.00      1.00    101672
weighted avg       1.00      1.00      1.00    101672

train accuracy: 1.0
Test accuracy: 1.0
```

```
In [29]: result = clf.score(x_test, y_test)
         print("Accuracy: %.2f%%" % (result*100.0))

         Accuracy: 100.00%
```

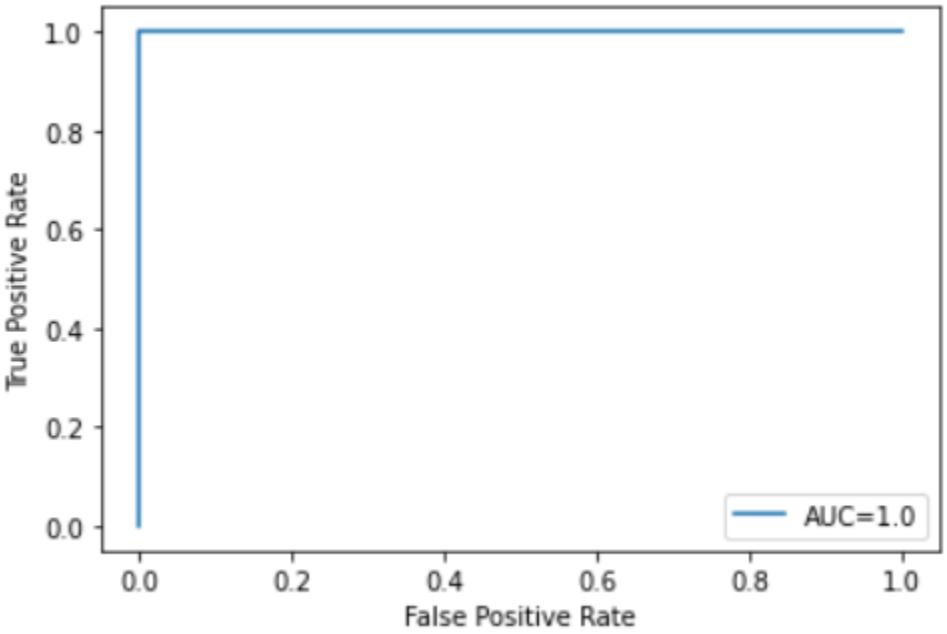
Accuracy of the model is 100%.

Below is a plot for False Positive Rate(X-axis) versus True Positive Rate(Y-axis) for a number threshold values between 0.0 and 1.0 .
The true positive is calculated as the number of true positives divided by the sum of the number of true positives and the number of false negatives. It describes how good the model is at predicting the positive class when the actual outcome is positive. True positive rate is also referred to as sensitivity.
The false rate is calculated as the number of false positives divided by the sum of the number of false positives and the number of true negatives.It is also called the false alarm rate as it describes how often a positive class is predicted when the actual outcome is negative.
The smaller values on the X-axis of the plot indicate lower false positives and higher true negatives.
Larger the values on the Y-axis of the plot indicates higher true positives and lower false negatives.
A perfect model represented at (0,1). A perfect model represented by a line that travels from bottom left of the plot to the top left and then across the top to the top right.

```
from sklearn import metrics
def printroccurve(y_test,  preds):
    fpr, tpr, _ = metrics.roc_curve(y_test,  preds)
    auc = metrics.roc_auc_score(y_test, preds)

 #create ROC curve
    plt.plot(fpr,tpr,label="AUC="+str(auc))
    plt.ylabel('True Positive Rate')
    plt.xlabel('False Positive Rate')
```

```
    plt.legend(loc=4)
    plt.show()
printroccurve(y_test,  preds)
```



A best-case ROC would look like a 90 degree angle. If you have this curve, then you probably don't need statistics. The curve shows that there are no false positives and no false negatives. The AUC of this ROC is 1.

Preprocessed holdout data is read into ‘testdataframe’ dataset

```
testdataframe=pd.read_csv('preprocessed_holdout.csv',low_memory=False)
```

Making the model predictions on the ‘testdataframe’ dataset

```
preds=clf.predict(testdataframe)
#merging input data with prediction
testdataframe['covid_vaccination'] = preds
```

```
testdataframe.head(5)
```

	Unnamed: 0	Unnamed: 0.1	ID	auth_3mth_post_acute_dia	rx_gpi2_
0	0	0.000000	0.230887	1.0	
1	1	0.000002	0.022477	1.0	
2	2	0.000004	0.046047	1.0	
3	3	0.000006	0.510482	1.0	
4	4	0.000008	0.176064	1.0	

5 rows × 368 columns

```
testdataframe['covid_vaccination'].value_counts()
```

```
0    355705
1    169453
Name: covid_vaccination, dtype: int64
```

Finally, the holdout dataset is appended with the target variable ‘covid_vaccination’ and now we can evaluate the model performance by comparing the target variable values with the original holdout dataset with the covid_vaccination variable that the Humana company holds for the competition.

```
testdataframe.to_csv("xgboost_holdout.csv")
```

Finally the testdataframe dataset is converted to ‘xgboost_holdout.csv’ file.

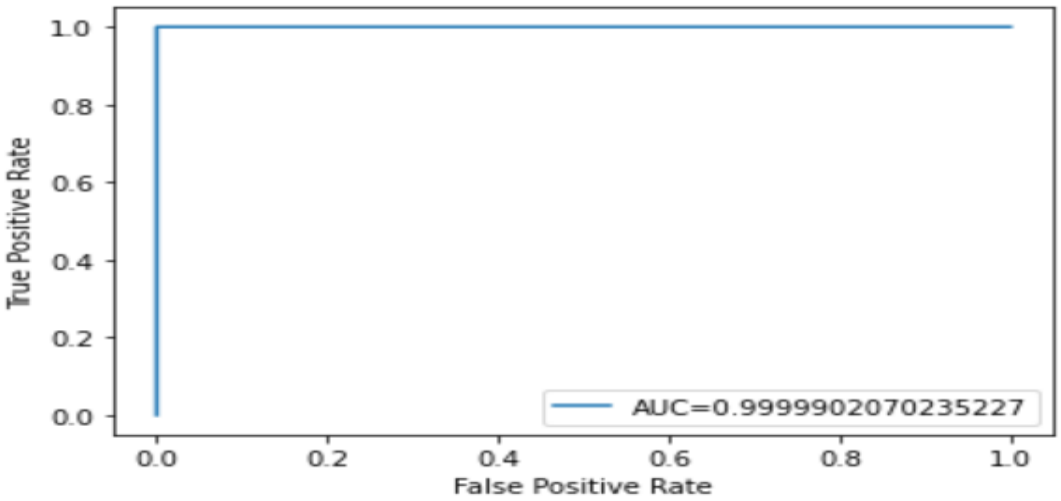
3.2. SVM

Evaluate model performance

```
#Import svm model
from sklearn import svm

#Create a svm Classifier
clf = svm.SVC(kernel='linear') # Linear Kernel

printroccurve(y_test,  preds)
```



The AUC of this ROC is 0.99

Preprocessed holdout data is read into the ‘testdataframe’ dataset and makes the model predictions on the ‘testdataframe’ dataset. Finally, the holdout dataset is appended with the target variable ‘covid_vaccination’.

```
testdataframe['covid_vaccination'].value_counts()
```

```
0     355712
1     169446
Name: covid_vaccination, dtype: int64
```

```
testdataframe.to_csv("svm_holdout.csv")
```

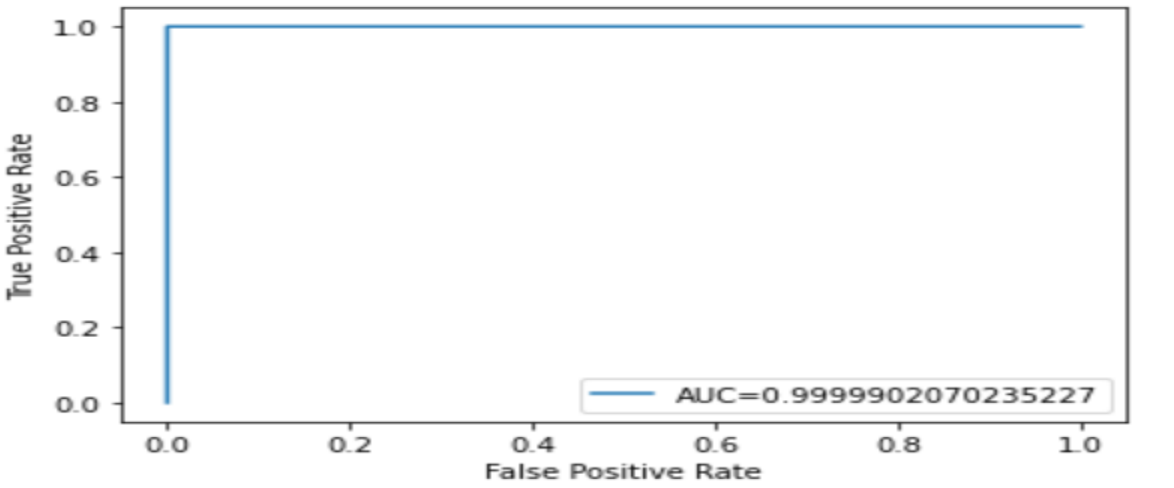
3.3. RANDOM FOREST

The Random Forest consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes a model prediction.

The low correlation between the models is the key of a random forest. The reason for this effect is that the trees protect each other from their individual errors. While some trees may be wrong, many other trees will be right, so as a group the trees are able to move into the correct direction.

So in a random forest we get the trees that are not only trained on different sets of data but also use different features to make decisions.

```
printroccurve(y_test,  preds)
```



The AUC of this ROC is 0.99.

Preprocessed holdout data is read into the ‘testdataframe’ dataset and makes the model predictions on the ‘testdataframe’ dataset. Finally, the holdout dataset is appended with the target variable ‘covid_vaccination’

```
testdataframe['covid_vaccination'].value_counts()
```

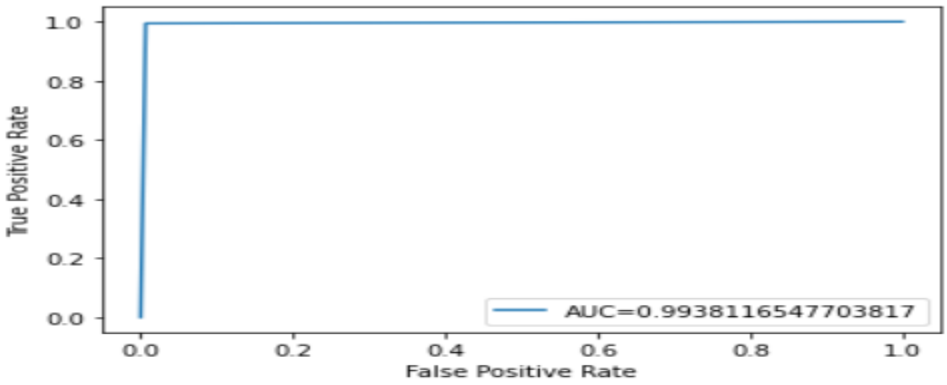
```
0    355730
1    169428
Name: covid_vaccination, dtype: int64
```

```
testdataframe.to_csv("randomforest_holdout.csv")
```

3.4. LOGISTIC REGRESSION

Logistic Regression is a statistical model that uses a logistic function to model a binary dependent variable. In regression analysis, logistic regression is estimating the parameters of a logistic model. Binary logistic model has a dependent variable with two possible values, such as pass/fail which is represented by an indicator variable, where the two values are labelled as “0” and “1”. In the logistic model, the log-odd for the value labeled “1” is a linear combination of one or more independent variables(“predictors”); the independent variables can be binary variables or a continuous variable.

For Example: To predict whether a person is hesitant to take vaccine (1) or (0).



The AUC of this ROC is 0.99.

```
testdataframe['covid_vaccination'].value_counts()
```

```
0    375803
1    149355
Name: covid_vaccination, dtype: int64
```

```
testdataframe.to_csv("logisticregression_holdout.csv")
```

Preprocessed holdout data is read into the ‘testdataframe’ dataset and makes the model predictions on the ‘testdataframe’ dataset. Finally, the holdout dataset is appended with the target variable ‘covid_vaccination’

3.5. DECISION TREE

Decision Trees are a class of very powerful Machine Learning model capable of achieving high accuracy in many tasks while being highly interpretable.This structure holds and displays the knowledge in such a way that it can easily be understood, even by non-experts.

Steps to build a tree:

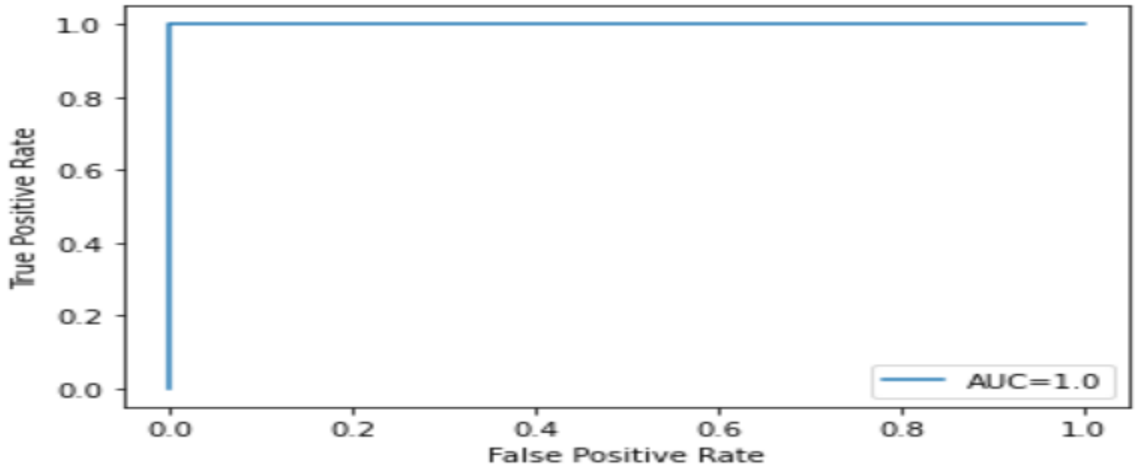
Begin with your training dataset, which should have some feature variables and classification or regression output.

Determine the “best feature” in the dataset to split the data on; more on how we define “best feature” later.

Split the data into subsets that contain the possible values for this best feature. This splitting basically defines a node on the tree i.e each node is a splitting point based on a certain feature from our data.

Recursively generate new tree nodes by using the subset of data created from step 3. We keep splitting until we reach a point where we have optimised, by some measure, maximum accuracy while minimising the number of splits / nodes.

```
printroccurve(y_test,  preds)
```



The AUC of this ROC is 1.0.

```
testdataframe['covid_vaccination'].value_counts()
```

```
0    355705
1    169453
Name: covid_vaccination, dtype: int64
```

```
testdataframe.to_csv("decisiontree_holdout.csv")
```

Preprocessed holdout data is read into the ‘testdataframe’ dataset and makes the model predictions on the ‘testdataframe’ dataset. Finally, the holdout dataset is appended with the target variable ‘covid_vaccination’

CONCLUSION:

As we got AUC as 1 for two models, XGBOOST and Decision Tree. The model is able to clearly generalize the training and test sets. We use those models on the holdout data to identify the people that are least hesitant to vaccine and target them by making the vaccine more available. Those who are highly hesitant to take the vaccine are educated and provided other attractive offers for taking the vaccine. These types of solutions reduce the cost of handling and promoting vaccination drives. It also allows the vaccine production companies to plan their production as per the demand of the public such that overly produced vaccines are not wasted. Our final models are XGBoost and Decision Trees which have given an AUC of 1. The rest of the models such as Linear Regression, Random Forests and SVM have given 99.3-99.9 % accuracy in identifying the target based on the predictor variables.

REFERENCES:

https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d

https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classification-in-python/

https://scikit-learn.org/stable/modules/tree.html

http://mays.tamu.edu/humana-tamu-analytics/wp-content/uploads/sites/113/2021/10/top-5.pdf

https://towardsdatascience.com/understanding-random-forest-58381e0602d2