Model Training for Flu Shot Learning

Kaan Efe Keleş Mehmet Eren Bulut Murat Şahin

Abstract

In this project, we dealed with some features of people in order to predict whether if a person got H1N1 vaccine and seasonal vaccine. We analyzed data, tried different techniques and tested different models. Finally, we obtained a successful model to predict what is asked.

1. Introduction

With the COVID-19 pandemic going on, importance of vaccines has been realised. Vaccines provide immunization for individuals, and enough immunization in a community can further reduce the spread of diseases through ”herd immunity.” Which makes predicting if a person got vaccinated a quite relevant problem in todays age.

2. Project description

Flu Shot Learning competition is hosted by DrivenData. They aim to answer, how general features of people are associated with personal vaccination patterns which can provide guidance for future public health efforts.

Contestants needed to predict how likely individuals are to receive their H1N1 and seasonal flu vaccines using machine learning technique.

3. Methodology

3.1. Data Analysis

The provided data was result of a survey that conducted about participants certain characteristics and whether they received H1N1 flu vaccine or seasonal flu vaccine. Some of the obtained **features** that we will refer in this section:

- h1n1_concern: Level of concern about the flu.
- h1n1_knowledge: Level of knowledge about the H1N1 flu.
- behavioral_outside_home: Has reduced contact with people outside of own household.
- doctor_recc_h1n1: H1N1 flu vaccine was recommended by doctor.
- doctor_recc_seasonal: Seasonal flu vaccine was recommended by doctor.
- opinion_h1n1_vacc_effective - Respondent’s opinion about H1N1 vaccine effectiveness.
- opinion_h1n1_risk - Respondent’s opinion about risk of getting sick with H1N1 flu without vaccine.
- opinion_h1n1_sick_from_vacc - Respondent’s worry of getting sick from taking H1N1 vaccine.
- opinion_seas_risk - Respondent’s opinion about risk of getting sick with seasonal flu without vaccine.
- opinion_seas_sick_from_vacc - Respondent’s worry of getting sick from taking seasonal flu
- household_children - Number of children in household, top-coded to 3.

Target values:

- h1n1_vaccine - Whether respondent received H1N1 flu vaccine.
- seasonal_vaccine - Whether respondent received seasonal flu vaccine.

Target and attribute analysis:

Prior to begin the training process, the information obtained by the survey is analyzed.

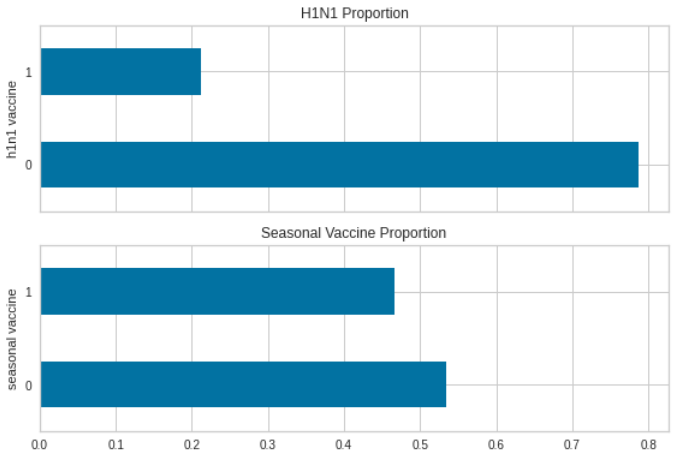

Figure 1. Ratio of the participants who receive the H1N1 flu vaccine and seasonal flu vaccine.

It can be observed from Figure 1 that only 20% of the participant receive H1N1 flu vaccine, whereas seasonal flu

vaccine is received by nearly half of the participants.

Attributes are analyzed in order to have an idea on how to train the model. Following charts explains the observations.

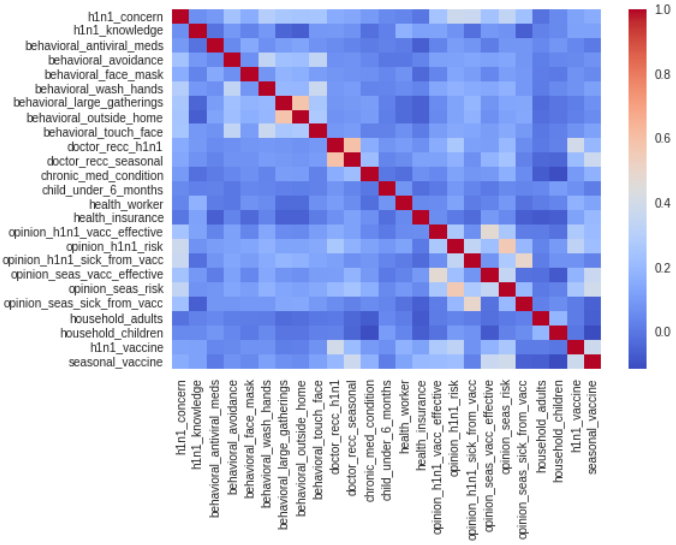

Figure 2. Heatmap for the features and targets in the dataset. The correlation increases as proceeded from blue to red.

The heat map in Figure 2 shows the correlation between each feature. Due to the fact that the heatmap mainly consists of blueish colors, we can infer that the attributes are usually independent.

Target values h1n1_vaccine and seasonal_vaccine has approximately 0.4 correlation. This shows that target values are not completely independent.

It can also be seen that there are some correlation between the following features:
"opinion_h1n1_risk" and "opinion_seasonal_risk"
"doctor_rec_h1n1" and "doctor_rec_seasonal"
"opinion_seas_risk" and "opinion_h1n1_risk"

Even there are some correlation, these correlations seems too weak. Therefore, most probably, dimensionality reduction or feature selection will not be significantly beneficial. Nevertheless these methods will be tried in the model for somewhat correlated features.

In the Figure 2 it can be seen that there is approximately 0.4 correlation between doctor_recc_h1n1 and h1n1_vaccine. This correlation is visualized in Figure 3. The ratio of H1N1 is dependent on whether their doctor recommended the H1N1 vaccine. If the doctor did not recommend it, people most likely not receive the H1N1 vaccine. On the other hand, people are likely to receive the H1N1 vaccine if their doctor proposes it.

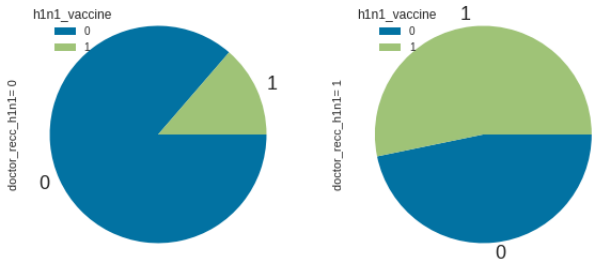

Figure 3. Ratios of participants that receive H1N1 vaccine or not with respect to whether their doctor recommended the H1N1 vaccine.

The Figure 4 visualizes the opposite case that there is almost no correlation between seasonal_vaccine and household_children.

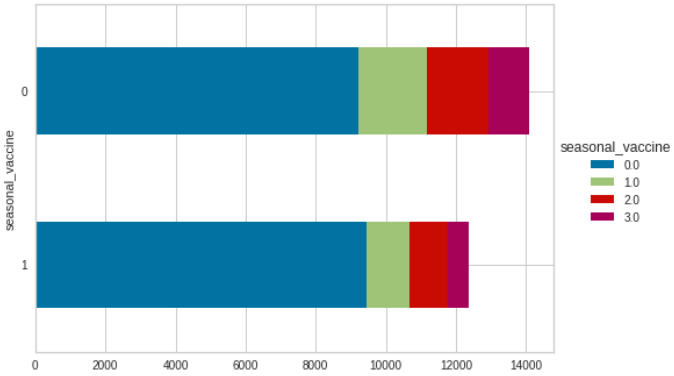

Figure 4. Whether the seasonal vaccine is received with respect to the number of children in the household.

Each proportion of the number of children remains relatively same whether seasonal vaccine received or not. Removing these kind of attributes could result in improvement on the model. However as we will see on the following sections it is not the case.

3.2. Preprocessing

After our data analysis, it was obvious to us that we must preprocess our data heavily. Numeric values were not scaled, null values were prevalent and we had to find a strategy to deal with categorical and ordinal features. We tried many methods of imputation and scaling to improve our model. Our outcomes are quite surprising for numeric features.

3.2.1. Encoding and Normalizing

Categorical and ordinal features
We applied one-hot encoder provided by the scikit-learn library to encode out categorical and ordinal features to a one-hot numeric array.

Numeric features
Numeric features were in dire need to be scaled, some of the features were either zero or one while many ranged over zero to five, which gave them an unfair leverage at manipulating the model. We tried many scaling options provided by scikit-learn library mainly:

- **StandardScaler:**
  Standardizes features by removing the mean and scaling to unit variance.
- **MinMaxScaler:**
  Transform features by scaling each feature between zero and one.
- **MaxAbsScaler:**
  Scales each feature by its maximum absolute value such that the maximal absolute value of each feature in the training set will be one.
- **RobustScaler:**
  This scaler removes the median and scales the data according to the quantile range. Outliers can often influence the sample mean / variance in a negative way. In such cases, the median and the interquartile range often give better results.

We tested all four methods and found the most success with RobustScaler, we believe this is due to outlier weakening features of this scaler.

After we scaled our numeric features, we applied QuantileTransformer to curb the leverage of our outliers even more so. QuantileTransformer applies a non-linear transformation such that the probability density function of each feature will be mapped to a Gaussian distribution; it will also automatically collapse any outlier by setting them to the defined range boundaries zero and one.

Since we didn't have access to a lot of data, we chose not to do outlier selection and elimination, we were satisfied by the outlier weakening capabilities of RobustScaler and QuantileTransformer.

3.2.2. Imputation

Our data was riddled with null values, which we thought we needed to impute somehow. We tried many methods of imputation and combinations of different imputation methods on different features. All of which we implemented using scikit-learn's provided SimpleImputer and IterativeImputer.

Categorical and ordinal features
We completed our data for categorical and ordinal features by imputing the most frequent value to our missing data.

Numeric features
We tried imputing mean, median and interpolation of values to complete our numeric data. Interpolation was very slow, performed the worst while the mean performed the best.

IterativeImputer
IterativeImputer is an experimental feature provided by scikit-learn, it is so experimental that in order to use it you have to first import experimental features from scikit-learn. Basically it is a multivariate imputer that estimates each feature from all the others. Sounds very advanced and useful but turned out to perform the absolute worst of all.

What did we end up doing? Disappointingly we ended up doing nothing. Our model performed the best when we just left out null values as is. See Figure 5. Scikit models handle null data so well that any imputation is just a bad guess.

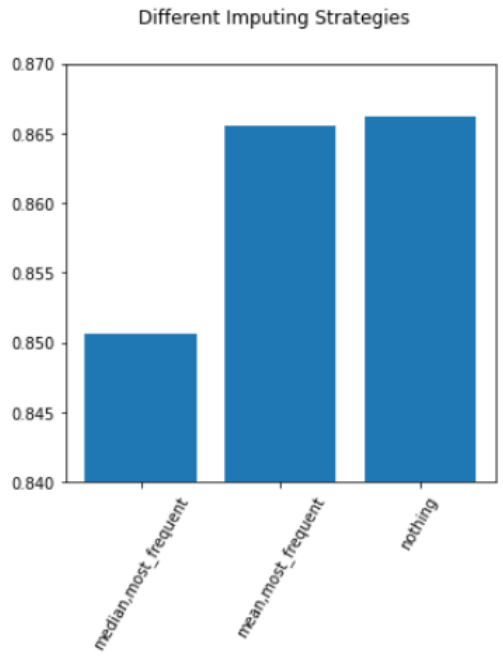

Figure 5. This graph shows different imputing strategies and their accuracies

3.3. Model Selection

We almost tried every single model in scikit-learn. We had instant success with ensemble models. We tried all the ensemble models and ended up using Histogram Gradient Classifier which performed the best. All linear models were sorely disappointing. Multi layer perceptrons(which we were hopeful about) took the longest to train with worst results. SVM was ok. Adaboost was promising but completely irrelevant after we tried other ensemble models.

The statistics in Figure 6 represent the averages of area under curve between h1n1_vaccine and seasonal_vaccine accuracies. For the models that accepted null values no

imputation was made, for the models that didn't we used our best imputation strategy which was taking the mean with the numeric values and taking the most_frequent with the categorical and ordinal values.

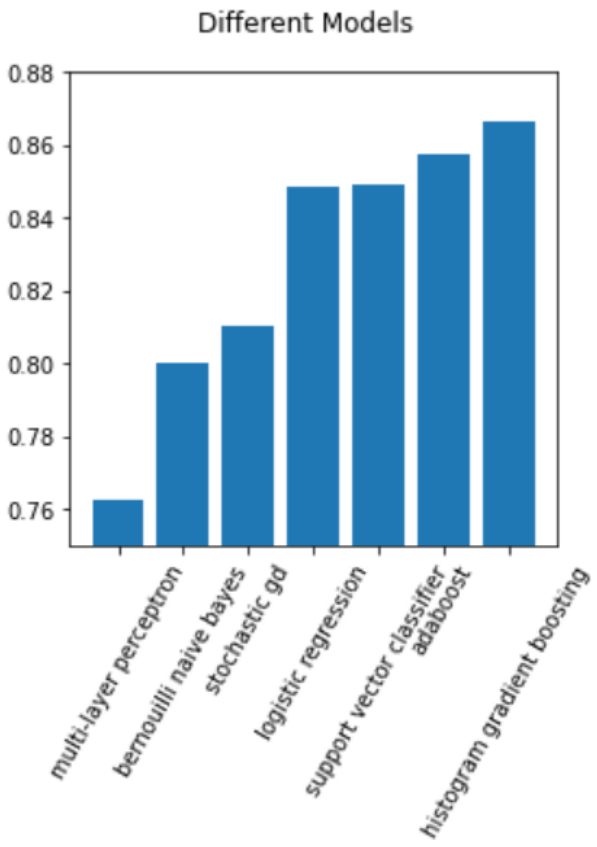

Figure 6. Average area under curve between both accuracies

3.4. Feature Selection

We tried to select the best features and eliminate the useless ones by using scikit-learn's VarianceThreshold. It calculates features' training-set variance and if it is lower than the given threshold then that feature is removed.

We tried this over many thresholds as can be seen on Figure 7. 0.5 gives the same results as 0 we can therefore say that no feature exists with training-set variance less than 0.5, increasing the threshold slightly tells a different story: the area under curve measurements dip significantly and keep getting worse as we increase the threshold, oddly there seems to be a slight increase at the threshold level 0.9.

It is observed that reducing the number of features is not profitable for our model. We believe this is due to our model having quite few features even if a feature is bad it still has some predictive power in our model and taking that away seems to make our model worse.

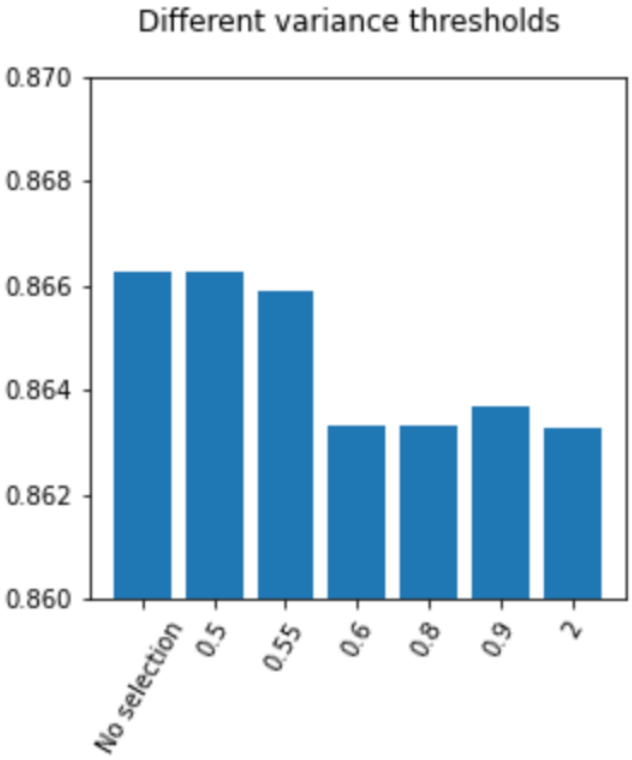

Figure 7. Accuracy with different variance thresholds. Number of features decreases as the variance threshold increases.

3.5. Model Training

After selecting our model, we trained two pipelines for our two objective functions. Both pipelines share the same preprocessing and scaling, with slightly different parameters on our estimator functions. We tried some parameter optimization with scikit-learn's GridSearchCV with disappointing results. We barely changed the default parameters, this is not for a lack of trying to optimize, scikit-learn's default parameters are just so good that the room for optimization is so little. We ended up using different learning rates for our pipelines, the regularization method of choice was L2 which we tuned the weight of and we turned off early stopping for our model.

4. Results

Model evaluation is calculated with AUROC. This metric computes the area under the Receiver Operating Characteristic (ROC) Curve which is a graph showing the performance of a classification model at all classification thresholds. The aim was maximizing the AUROC.

Our best model ended up scoring 0.8620 which got us ranked 129th out of 3202 competitors at the time of our submission which was late at night of our deadline. See Figure 8. The leading model scores 0.8658, which we are just %0.004 falling short from.

Figure 8. Our final submission to competition

5. Conclusion

A general idea was obtained about how a machine learning model is developed. We had hands-on experience with data analysis methods that are used to decide which algorithms to use. Numerous algorithms were tried and results were compared to improve the model. We gained familiarity with the technology stack used for such a project. Working as a team was a delightful experience with responsible and helpful teammates to help along the way.

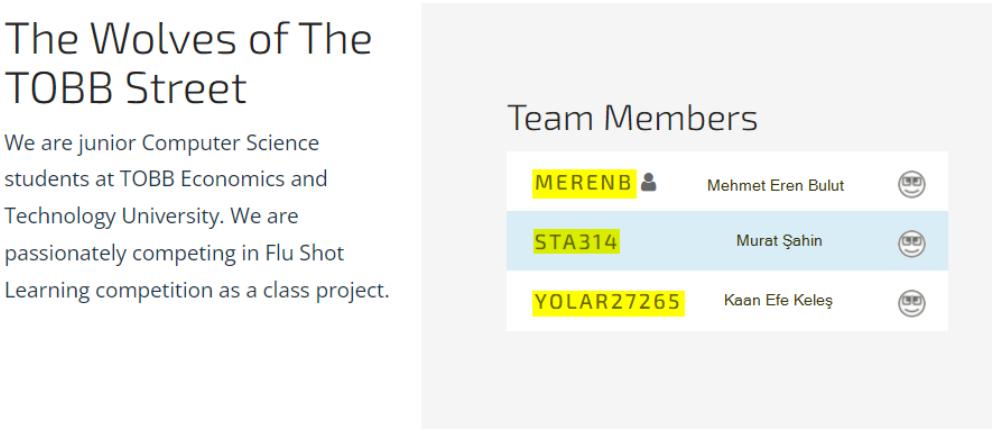

Figure 9. Our team