Flu Shot Learning – Predict H1N1 and Seasonal Flu Vaccines

Bachelor's Studies Seminar – Data Science Competition

Summer Semester 2025, Department of Statistics

Ludwig Maximilian University of Munich

August 26th, 2025

Nhi Nguyen

Matriculation Number: 12622884

Competition Name: nhi_nguyen

Vanilton Paulo

Matriculation Number: 12448685

Competition Name: awe6730

Supervisor: Giuseppe Casalicchio

Github Repository: https://github.com/nhi-nguyen04/Data-Science-Competition

Abstract

H1N1 and Seasonal Flu continue to pose public health challenges, with vaccinations being a critical preventative measure.
This project aims to predict individuals' vaccine uptake for H1N1 and Seasonal Flu using personal, behavioral, and opinion-based survey data from 2009, as part of a data science competition on DrivenData.org – Flu Shot Learning: Predicting H1N1 and Seasonal Flu Vaccines.

By utilizing the tidymodels framework in R, we develop several classification models, ranging from Logistic Regression to LightGBM models, to predict vaccine uptake based on the provided dataset.
The analysis began with an informative data exploration, data preprocessing, and feature engineering, which included steps such as imputing missing values, encoding categorical values, and creating domain-relevant interaction terms. A variety of models were then trained and evaluated using metrics such as ROC AUC.

Among all models tested, both Logistic Regression and LightGBM models demonstrated strong predictive performances out of all models that were trained and tested.

.

Contents

1. Introduction [Nhi Nguyen]

Understanding the factors that influence vaccine usage is important for public health planning, especially during pandemics. This project focuses on predicting how likely it is for an individual to receive the H1N1 and/or Seasonal Flu vaccine using a dataset provided by the DrivenData Flu Shot Learning Competition.

The dataset includes various types of features such as personal information, behavioral patterns, and health-related opinions, which will be the foundation for our predictive model. These models can potentially inform public health interventions by identifying groups with lower vaccination probabilities accordingly.

We aim to build and evaluate several different predictive models that estimate the probability of vaccine uptake using a variety of features.

This project applies a tidy and reproducible machine learning workflow using the tidymodels ecosystem in R. Various models – including Logistic Regression, Random Forest, and LightGBM – are trained, tuned and finally evaluated.

Through thorough data preparation, Cross Validation (CV), and evaluation metrics like Receiver Operator Characteristic (ROC) and Area under the Curve (AUC), we aim to produce models that are both effective and insightful.

Beyond the technical modeling, the report also introduces ideas that can be considered in the future while keeping this project in mind.

2. Data Overview [Nhi Nguyen]

2.1 Data Collection

The data for this project was collected back in 2009 and 2010, when there was a pandemic caused by the H1N1 influenza virus. During this pandemic, an estimated 151.000 to 575.000 deaths globally in the first year have been documented. In October 2009, a vaccine for the virus was released to the public. [1]

Around late 2009 and early 2010, a phone survey was conducted by the National H1N1 Flu Survey (NHFS) to find out whether an individual had received the H1N1 and/or Seasonal Flu vaccines. Additionally, the people were also required to answer questions about personal life, health behavior and opinion on the Flu as well as vaccines. [2]

This phone survey followed the principles of a random digit dialing telephone survey of different households monitoring the influenza immunization coverage in 2009 to 2010, which means the information was sampled using randomly generated phone numbers rather than selecting them from an already existing phone book. This ensures a more representative sample of the target population by including individuals that may not be listed in traditional directories. [3]

Households that were eligible for this survey and are therefore part of the target population are those that include people who are six months or older and lived in the United States during the survey.

2.2 Data Structure

The competition provides us with three relevant data sets – Training Features and Labels, as well as Test Features. We have a total of 26.707 observations and 36 features – one of them being the Respondent's ID – to work with. The feature variables range from numeric, categorical, and binary with the majority being either binary or categorical.

The Training Labels consist of the Respondent's ID as well as the vaccination statuses to H1N1 and Seasonal Flu vaccine. In order to match the information of an individual's personal life, health behavior, and opinion on the flu with the vaccine status, we use the Respondent ID.

2.3 Target Variables

We have two target variables – received H1N1 and/or Seasonal Flu vaccine.
Both outcomes are binary, meaning each target can have two labels: 0, which indicates not vaccinated, and 1, which indicates vaccinated.

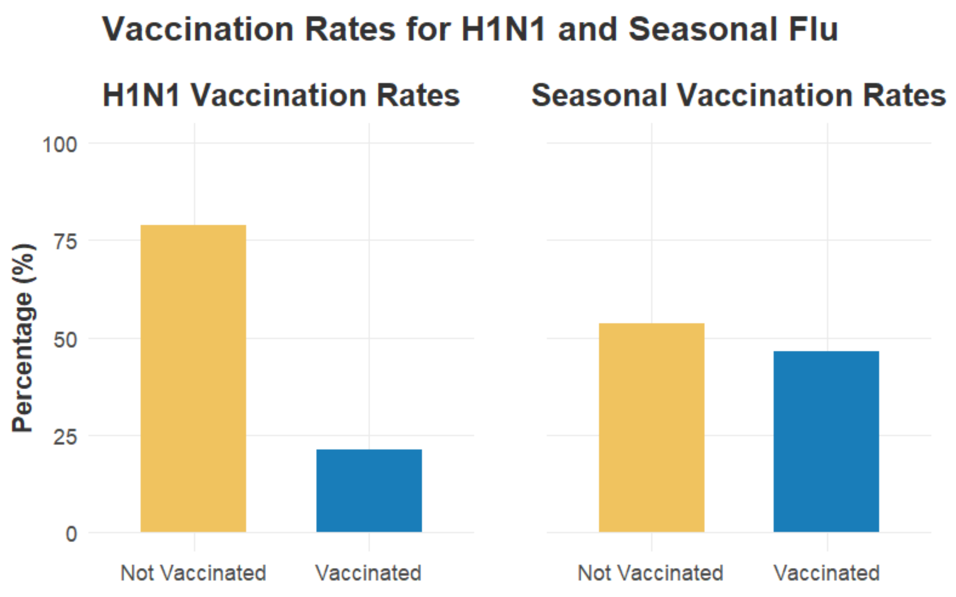

Fig. 1 Vaccination Rates for H1N1 and Seasonal Flu

The majority of individuals who participated in this survey did not get the H1N1 vaccination. In contrast, the vaccine distribution for the Seasonal Flu is more balanced, with both statuses reaching values around 50%.

From a class distribution perspective, the H1N1 vaccine target shows a class imbalance, which means one vaccination status is heavily outnumbering the other. This problem can make it difficult for the future model to accurately learn and predict the minority class. Additionally, the model may default to predicting the majority class, which can result in poor performance on the less common – but potentially more important – cases. [4]

In contrast, the Seasonal Flu vaccination status shows balanced classes, which means that the model will have a roughly equal number of examples for each outcome, making it easier for the model to learn both cases more effectively.

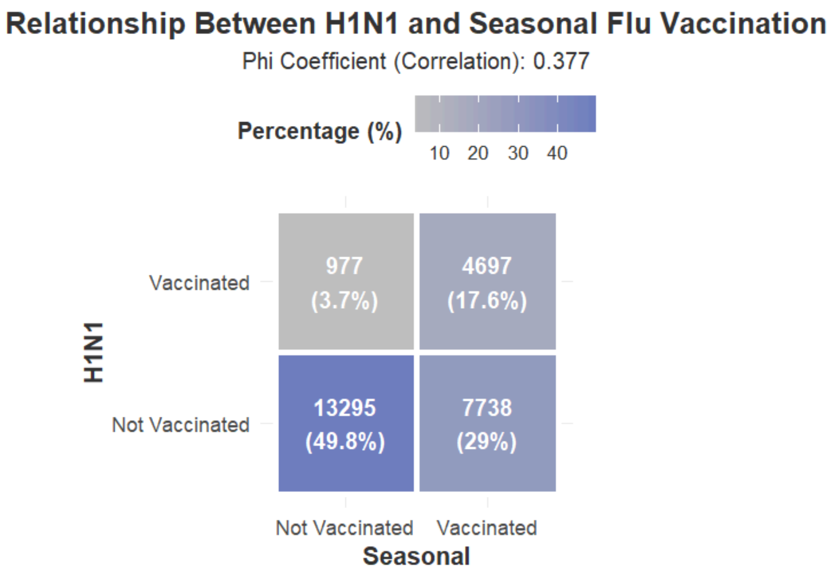

Fig. 2 Heatmap of H1N1 and Seasonal Flu Vaccination

To understand the relationship between our target variables, we create a heatmap, portraying the correlation of the two. Using this visualization we can determine the phi coefficient Φ to measure the association between these binary variables.

A value close to one indicates a positive correlation for the variables whereas a negative one shows a negative relationship. If the coefficient is close to zero, it can be assumed that there is no systematic pattern for them. [5]

That means our phi coefficient of 0.377 indicates a somewhat moderate relationship between H1N1 and Seasonal Flu vaccine. Knowing this, we model both outcome variables separately and independently from each other, which means neither of them will be used as a feature for the model.

2.4 Features

Our features contain various data types, such as numeric, categorical as well as binary.

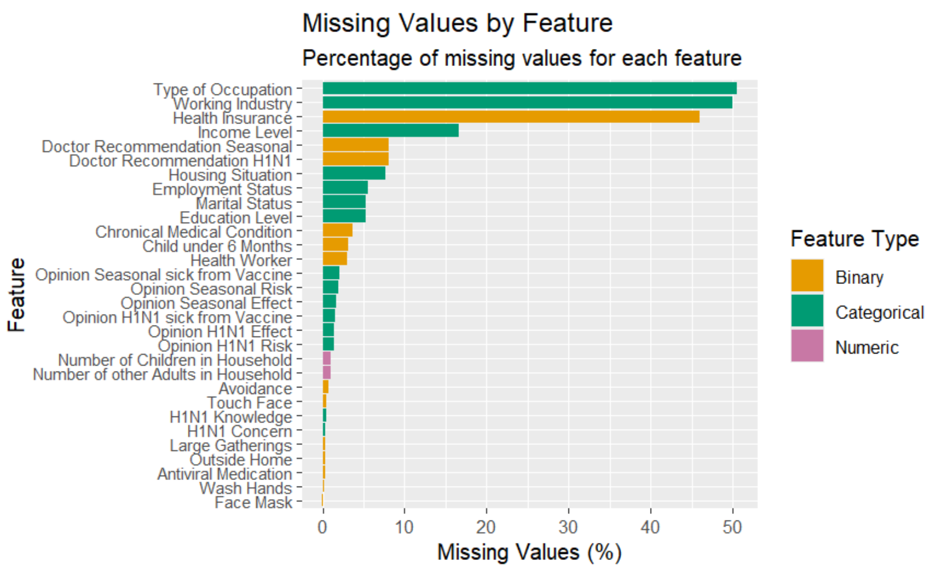

Fig. 3 Missing Value Distribution by Feature

Most features are missing observations under 10%. A few features – Type of Occupation, Working Industry and Health Insurance – have almost 50% of values absent.

By conducting a Little's missing completely at random (MCAR) test using the naniar package in R, we can find out whether the absent data at hand is missing with no patterns or not. If the result from the test is greater than the level of significance (here: 0.05) then the null hypothesis, which says that the data is MCAR, cannot be denied. [6] The test resulted in a value of 0.996 which is far greater than the level of significance and our missing data is therefore MCAR.
This now allows us to impute missing values which gives us unbiased estimates and standard errors. [7]

We now take a closer look on how certain features influence the vaccine intake. As we have more than 30 variables at hand, we limit them to four for each outcome.

We take a closer look at the H1N1 vaccination rate by the H1N1 concern, H1N1 knowledge, opinion on its effectiveness and age group.

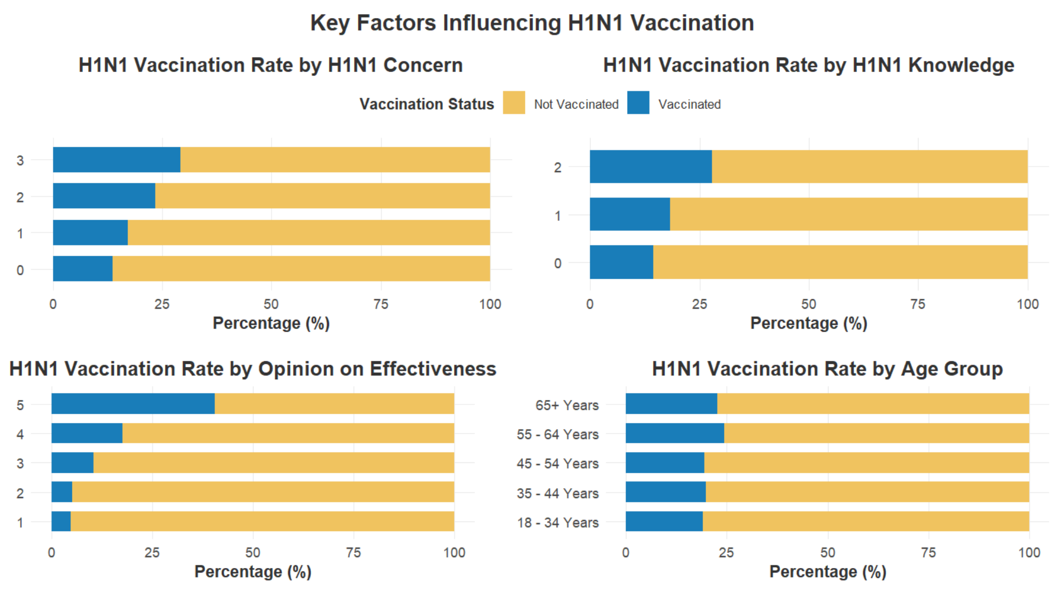

Fig. 4 H1N1 Vaccination Rate based on different Features

The lower the number on the x-axis, the lower the concern, knowledge and opinion on the effectiveness. With this in mind, we can interpret the plot appropriately. It is visible that the vaccination rates for every feature consistently show the majority as non-vaccinated. The same interpretation can also be seen in the Vaccination Rates plot (see Fig. 1).

What's noticeable is that with an increasing concern, knowledge and opinion on its effectiveness, the rates for vaccinated individuals grow in value as well.

For the Seasonal Flu vaccination rates, we look at the H1N1 concern, opinion on its risk, age group and education.

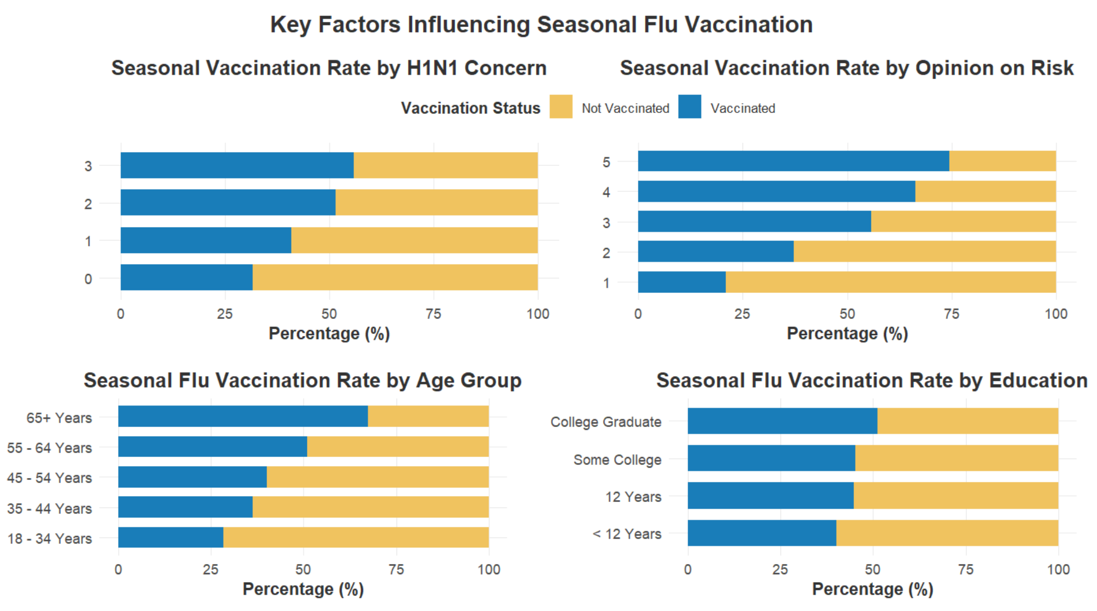

Fig. 5 Seasonal Flu Vaccination Rates by different Features

These demographic features have higher vaccination rates than the H1N1 vaccine rates. Knowledge and opinion questions seem a strong indicator for both target outcomes.

For Seasonal Flu, vaccination rates of older individuals, who face higher risks of flu-related complications, were higher. The H1N1 vaccine showed that, despite older adults being at a higher risk of complications, they were less likely to get infected, and the vaccination rate reflected this lower perceived risk.

3. Flu Shot Learning Competition [Vanilton Paulo and Nhi Nguyen]

3.1 Challenge Description

This competition takes a closer look at two types of vaccinations, which are both important public health measures to reduce the further spread of diseases.

The challenge for this contest is to find out whether we can create a model that can accurately predict the vaccination status for both target variables – whether someone has received the

H1N1 and/or Seasonal Flu vaccine or not based on information about their personal life, health behavior and opinions.

Considering this objective and the data at hand, a Supervised Machine Learning approach is ideal. As we are dealing with an issue where we have to predict two different labels, we compare multiple Classification methods and their performance in predicting the correct labels.

All models are being evaluated using the Area under the Curve (AUC) as specified by the competition. The final score that will determine our ranking on the public leaderboard of this contest is computed by using the mean value of the AUC for both the H1N1 and Seasonal Flu predictions. [8]

3.2 Working Tools

In this project we apply a variety of working tools.
We utilize the platform Github to keep track of our tasks and codes. It is mainly to keep our work structured and organized.

All the coding and execution was performed using R Studio and CIP servers for faster running time.

3.3 Methodology

3.3.1 Modeling Approach

For the modeling approach we make use of the tidymodels package in R.
The tidymodels framework is a collection of different R packages which are all applicable for modeling and machine learning purposes. [9]

It contains the packages rsample, recipes, parsnip, yardstick, broom, workflows, tune and dials.

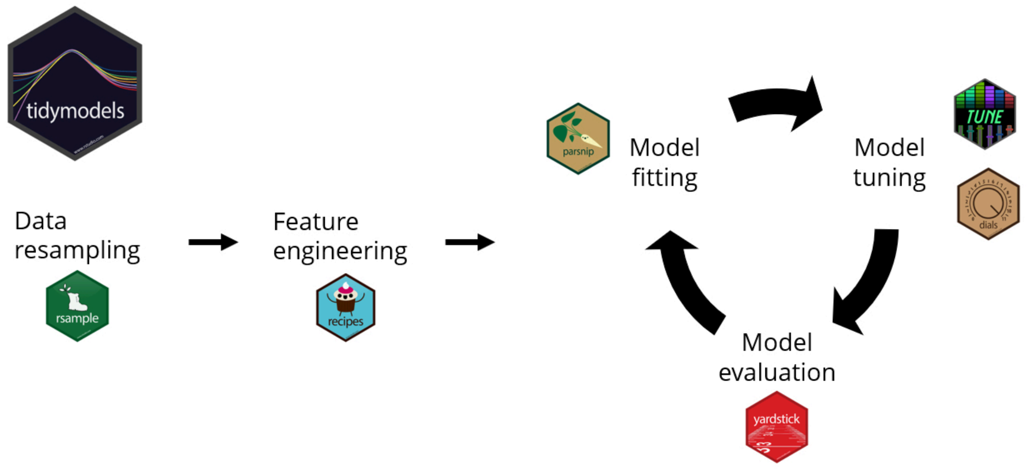

Fig. 6 Chen Xing, "Tidymodels Ecosystem Tutorial"

We start off by using the Rsamples package to split our Training Data into training and test sets. We then use the training set to perform feature engineering with the Recipes package. That means we prepare the given features in a way that we can apply them to the modeling. This includes steps like creating interaction terms. After finalizing the feature engineering steps, the actual model can be fitted using the Parsnip package. Using the Workflow package, we put everything into a compact workflow to keep track of our recipes and model specifications. This workflow will then be used to train the model using the training set from the initial split. That resulting model is utilized to predict labels for the testing set from the earlier split. These predicted classes will be compared pre-tuning to the original labels and evaluated using confusion matrices and Receiver Operator Characteristic (ROC) curves.
To improve the model's performance on unseen data and overall predictive power even further, we apply Cross Validation and Hyperparameter Tuning. Finally, we select the best model using ROC AUC and perform a last fit on held out split. Lastly, we train the model on the full training data and make predictions on the test data [10]

3.3.2 Baseline Model

Following this model approach, we start by creating a base model using Logistic Regression.

This is a Supervised Machine Learning classification algorithm which predicts either discrete or categorical outcomes. It estimates the probability of a specific outcome – in this case vaccinated or not vaccinated.

We use this model because it is easy to implement, interpret, and train. It is also quick. Efficiency is important as we have a tight deadline. With this fast baseline model, we can easily experiment with different ideas and approaches without wasting much time. [11]

3.3.3 Working Pipeline

We begin by splitting our original Training Data into two sets – 80% training and 20% testing sets – two separate times to stratify for both of our target variables H1N1 and Seasonal Flu status. Stratifying ensures that our training and testing sets contain approximately the same distribution of the outcome variables. [12] The training set will then be used to create our workflow while the testing set will be needed to evaluate the model. This will give us an unbiased assessment of our model, as the testing set has not been touched or used until now – neither changed nor used for training. [13] For now the testing set will be put aside.

Now working with the training set only, we create two separate recipes for each target variable – H1N1 and Seasonal Flu Vaccine. We start off by specifying our model using the package parsnip – in this case Logistic Regression.
Using the recipes package, we perform feature engineering, that means we preprocess our features to then use them to fit our model. The first step is to remove the target variable that will not be predicted and then impute the missing values in the training set – numerical features with their respective median. Other unidentified entries will be changed from NA to "unknown", meaning it will create a new factor class "unknown" to use along with the already existing ones. This is to ensure that later steps in the code work just fine with no errors arising.
After dealing with the absent values we can one-hot encode all categorical variables, that means each category in a categorical feature becomes a separate binary feature. This ensures that every section can be treated independently [14] and we start engineering different interactions using domain knowledge.

Lastly, we remove all features that show zero-variance – predictors who only have one value and therefore do not contribute much to the model – to help the model distinguish between observations and we normalize the numeric variables, which means they are being put on the same scale. What this essentially is that all numeric features will be changed to be in a similar range so we can compare them more easily.

After finalizing the recipes, we bundle everything – the feature engineering and model specification – into a clean and compact workflow using the workflows package. This allows us to keep track of everything more easily. This workflow can now be used to train on the training set and then evaluate the model's performance on the testing set, which were both created earlier.

To assess how well the model performs pre-tuning, we now predict the class probabilities and labels on the testing set. We use the ROC AUC as specified by the competition.

We want to maximize the score and improve our model's predictive power on unseen data to overall avoid overfitting. Using a method called Cross Validation (CV), we can greatly reduce the chances of our model learning the pattern of the training set too well. It also provides us with multiple estimates of model performance, allowing us to see if the model performs consistently well. [15]

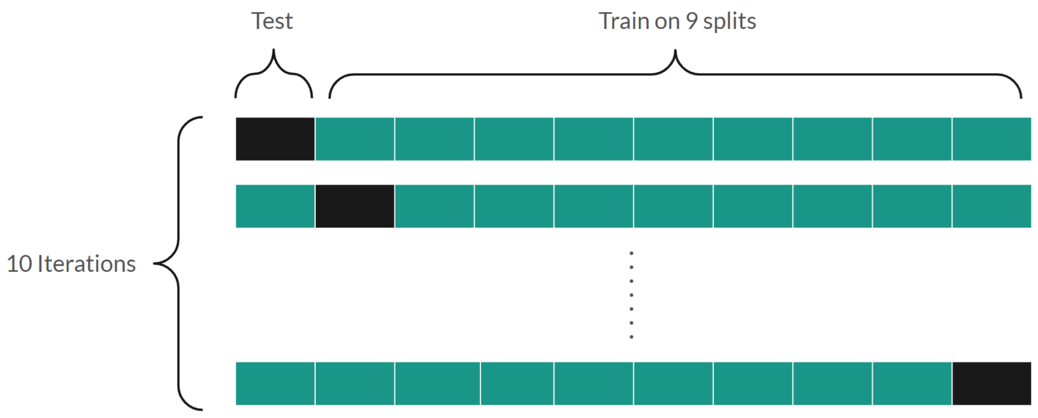

Fig. 7 Visual Representation of Cross Validation

We create 10 folds, respectively for the H1N1 and Seasonal Flu vaccine. That means the training set is randomly partitioned into 10 splits of roughly equal size which are used to perform 10 iterations of model fitting and evaluations.
To train the model, an iteration uses 9 splits to train the model. The one fold that is left over will be used to assess the model's predictive power giving us an independent estimate. This process is going to be repeated 9 more times, each time using different folds to train and evaluate the model. In the end, we should have 10 different estimates of the model's performance.

Using Cross Validation, we continue improving the model by finding the optimal set of hyperparameter values, also called Hyperparameter Tuning. We aim to balance the variance-bias tradeoff and minimize the loss function of the model to ultimately improve the overall performance.
We use Random Grid for this process, that means we generate random combinations of hyperparameter values for a grid search. This random sampling covers a wider range of values than a systematic selection and thus increases the chances of finding the optimal hyperparameter settings. [16]

At first, we used the default settings of the package for hypertuning.However, we were informed by our supervisor that this can be further optimized to achieve a higher ROC AUC score.

We specify our tuning space appropriately by using the mlr3 tuning spaces, which provides us with ready to use configurations for top Machine Learning algorithms. These spaces are based on peer reviewed research for broad dataset applicability and seamlessly integrates with our tidymodels framework. Our overall penalty strength λ has a space of s [1e - 04, 10000] log-scale and the mixing parameter between Ridge (0) and LASSO (1) has a space of Alpha [0, 1]. [17]

4. Model Performance and Evaluation [Nhi Nguyen and Vanilton Paulo]

4.1 Confusion Matrix

To properly judge and evaluate our model's performance, we first take a close look at its confusion matrices – for the H1N1 and Seasonal Flu Vaccine – pre-tuning, that means its performance before the hyperparameter tuning.

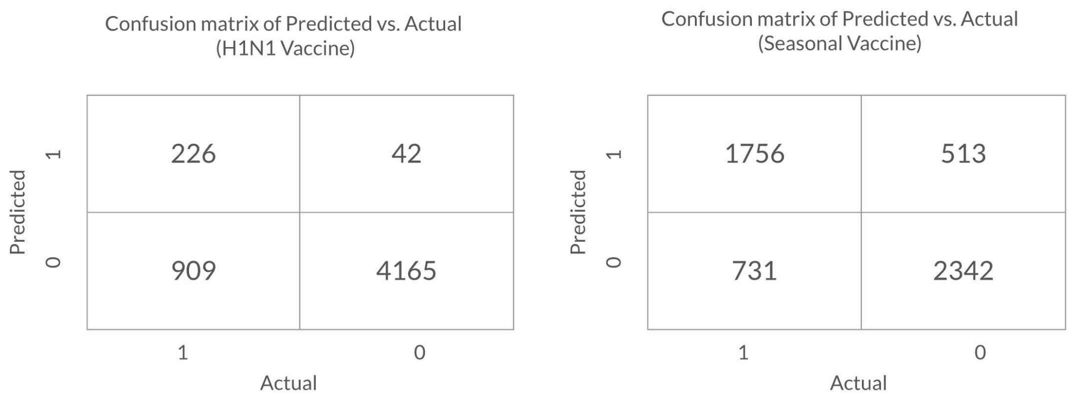

Fig. 8 Confusion Matrices for H1N1 and Seasonal Flu Vaccine

This matrix summarizes how well a classification model performs by comparing its predicted labels to the true labels. The True Positives (TP) – located in the top left corner – portray how many positive cases have been correctly classified as such, that means the amount of cases where a vaccinated person has been predicted as vaccinated. In contrast, the True Negatives (TN) – in the bottom right corner – show the correct predictions for the non vaccinated population. The visualization also presents cases that were falsely predicted – the False Negatives (FN) and False Positives (FP). The FN is located in the bottom left corner and shows how many cases are vaccinated but incorrectly classified. The FP is located in the top right corner and portrays the number of non-vaccinated people who were predicted as vaccinated. [18]

We can see in figure 8 that the amount of correctly predicted cases for the Seasonal Vaccine are relatively high. On the other hand the TP for the H1N1 vaccination are fairly low, which leads back to the class imbalance of H1N1 vaccination. As the majority of the population is not vaccinated, the model will have difficulties predicting the minority class – in this case the vaccinated. The model will default into predicting the majority class – the non vaccinated – which leads to the TN to be rather high and the TP to be on the smaller size.

4.2 Receiver Operator Characteristic and Area under the Curve

We can visualize the model's performance across different thresholds using the confusion matrix values – called a Receiver Operator Characteristic (ROC).

What this plot essentially shows is, the proportion correct among actual positives against the proportion incorrect among actual negatives across different thresholds as a step function. It is created by plotting the Sensitivity or True Positive Rate (TPR) against 1-Specificity or False Positive Rate (FPR). Using this curve we can compute the Area under the Curve (AUC) which will help us estimate the model's performance. The higher the AUC the better the predictive power. [19]

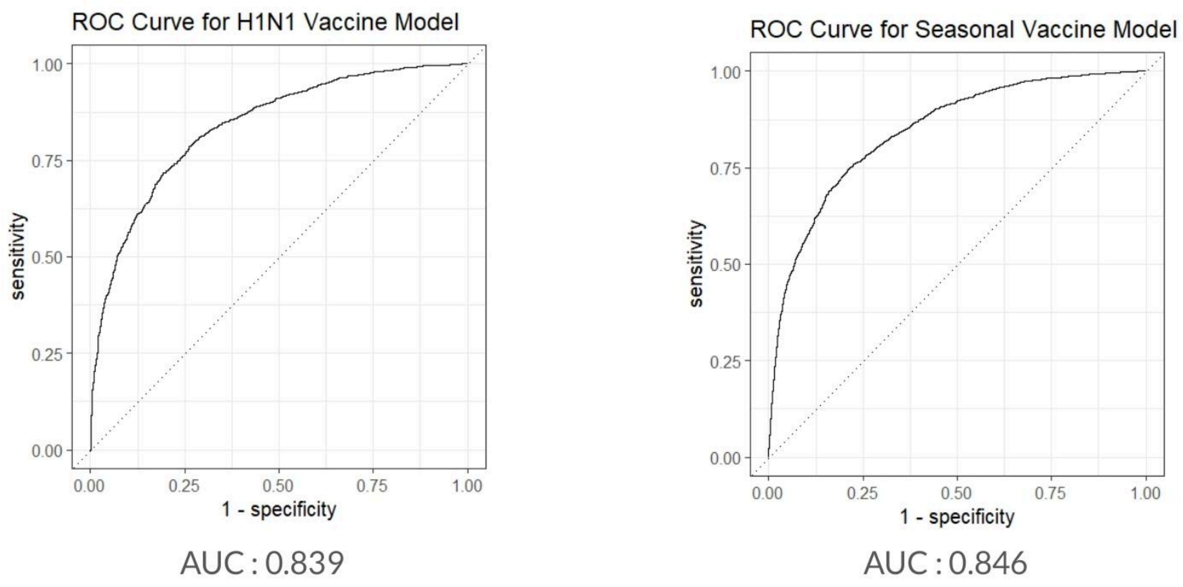

Fig. 9 ROC and AUC of H1N1 and Seasonal Flu models pre-tuning

Our ROC and AUC scores pre-tuning show relatively high values reaching over 0.8. For a pre-tuned base model that is very good.

To increase the AUC score we perform hyperparameter tuning on our Logistic Regression models.

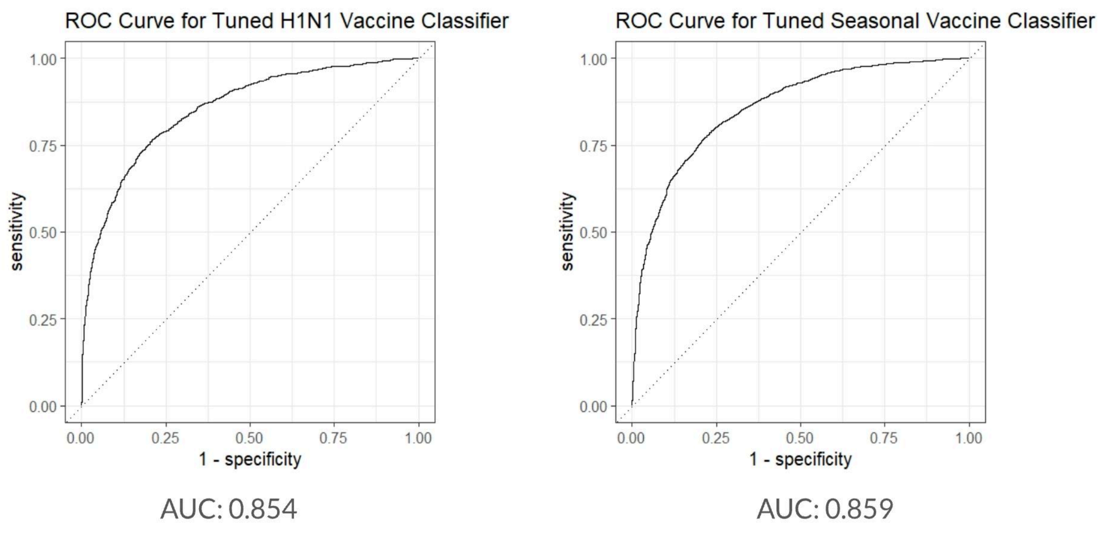

Fig. 10 ROC and AUC of H1N1 and Seasonal Flu models post-tuning

After tuning, we observed notable gains in our AUC scores. Our AUC increased by 0.015 and 0.013 for the H1N1 and Seasonal Flu models, respectively.

In theory, an AUC score of 1 would be perfect, which means the model performs perfectly and predicts everything accurately. There is a 100% probability that the model will correctly rank a randomly chosen positive example higher than a randomly chosen negative example. We aim for scores around 0.8-0.9, which indicates high predictive power. [20]

Once the tuning process is done, we select the best-performing model based on the ROC AUC. We do this separately for the two prediction tasks, H1N1 vaccine uptake and seasonal flu vaccine uptake.

The next step is to finalize the workflow by locking in the best hyperparameters. Now the model is ready for final training and evaluation. We evaluate the finalized model for the respective prediction task on a held-out test set(not seen during tuning). This will give an unbiased estimate of the model's performance. We collect the ROC AUC value from the holdout evaluation, and we visualize the ROC curves (see Fig. 10).

We then retrain on the full training dataset. Finally, we fully trained models are then used to predict on the test dataset - providing probability scores for both vaccine outcomes.

After submitting our predicted probabilities for each individual's vaccination status in the competition, we achieved a public score of 0.8542 and ranked 836th overall.

5. Experimentation and Comparison [Vanilton Paulo]

5.1 Other Models

We decided to discard other models, such as Decision Trees and Bagged Trees, in favor of the Random Forest because they had a lower ROC AUC score on both the locally and the public leaderboard. We also discarded XGBoost because it crashed often on the server, and it's very computationally expensive and difficult to iterate. We decided to use LightGBM for this reason.

We now want to further improve our score by expanding our model specifications to Random Forest and LightGBM. These are more tree-based models that work in a recursive manner partitioning the feature space. [21]

The recipe for feature engineering for the Random Forest differs from that of Logistic Regression, but both start by removing the vaccine that will not be predicted in the respective Flu model. We set Ranger as the engine.

We removed the one-hot encoding since Random Forest can handle factors directly. We included different types of imputation for missing values, including mode imputation for categorical variables and median imputation for numerical variables. Just like the Logistic Regression recipe, we added feature interactions based on our domain knowledge and removed predictors with zero variance; wrapped the model specification and recipe into a single workflow – separated for H1N1 and Seasonal Flu. Unlike Logistic Regression models, the Random Forest models do not require dummy [22] or normalized predictor variables. [23]

As we specified a different model, we also have to hyperparameter tune different parameters. For this model we take a closer look at the number of predictors that will be randomly sampled at each split when creating the tree models, the minimum number of data points in a

node that are required for the node to be split further and lastly the number of trees which are contained in the final ensemble. We use a Random Grid with a size of 500.
We specify our tuning space appropriately by using the mlr3 tuning spaces, which provides us with ready to use configurations for top Machine Learning algorithms.

We explored three key parameters: mtry, trees, and sample.fraction. The mtry parameter is defined within the range [0,1], which controls how many features are randomly selected at each node split. The trees parameter defined within the range [1,200] determines the number of trees in the ensemble. The parameter sample.fraction defined [0.1,1] controls the proportion of training data used to build each tree. By defining these ranges with a random grid , our goal is to identify hyperparameter combinations that will yield model configurations that can be then selected based on its ROC AUC values. [17] [24]

Following hyperparameter tuning, the subsequent steps in the workflow remain consistent with those outlined previously.

Unlike Random Forest, LightGBM follows a rather similar recipe for feature engineering to the Logistic Regression models. It starts by removing the vaccine feature that will not be predicted in the respective Flu model. In a further step, we continue by imputing missing numerical values with their respective median and then move over to one-hot encoding the categorical features. This is then followed by some interaction terms, again based on domain knowledge. Lastly, all features with a zero-variance will be removed as the same reason as for Random Forest applies. Everything will be put in a workflow yet again.
For the LightGBM hyperparameter search, we switched over to finetune's racing method, which evaluates candidates on batches of resamples and eliminates underperformers early. [9] We tune a total of four parameters, consisting of the number of trees, the tree depth, the learning rate and finally the sample size which contains over 500 random configurations.

While both models differ in recipes and workflow, the working pipeline stays the same. That means, we train the model on our splitted training data and evaluate its performance pre-tuning and then continue working our way up with CV and hyperparameter tuning.

5.2 Final Model Comparison

The last step is to finally compare all models using the ROC and AUC and decide which one has performed the best out of all fitted models.

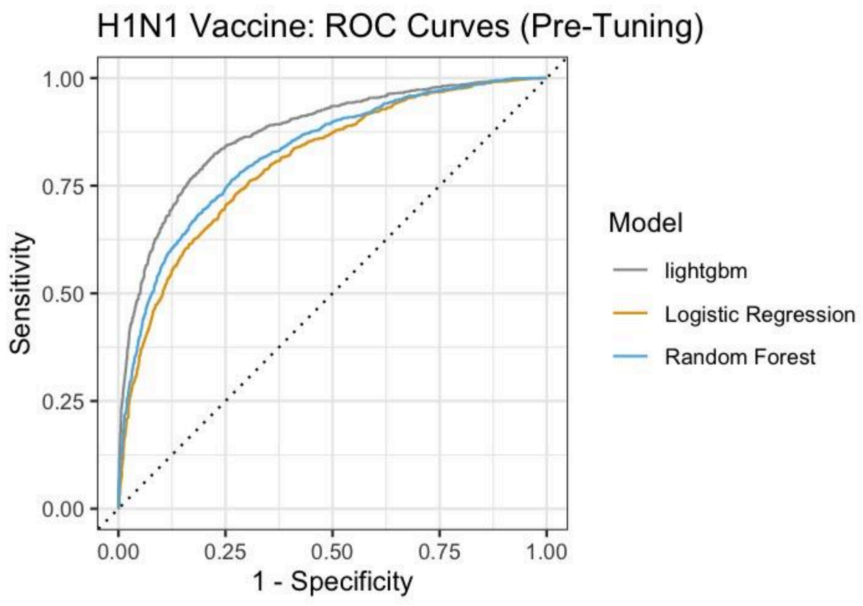

Fig. 11 Model Comparison of pre-tuning ROC curves for H1N1 Flu models

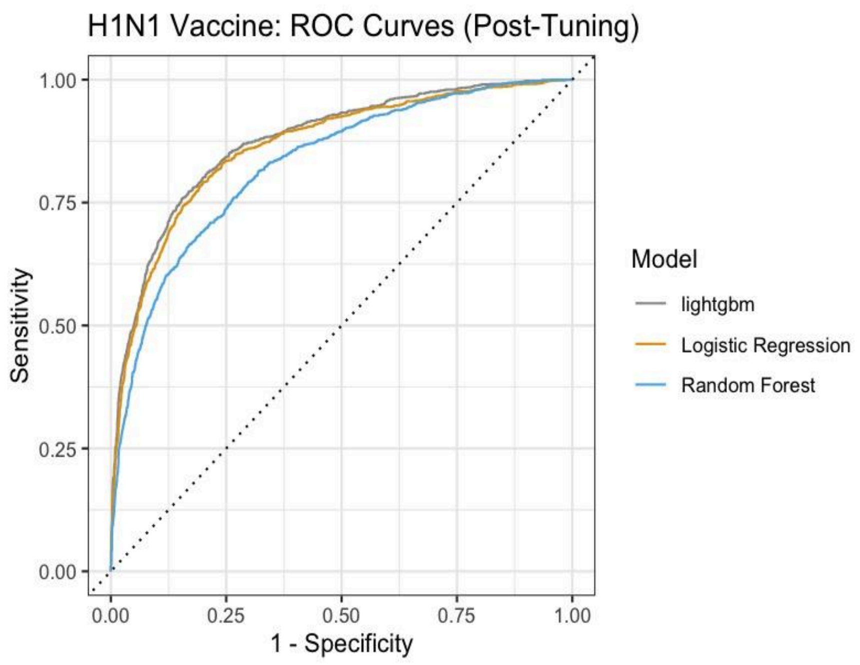

Fig. 12 Model Comparison of post-tuning ROC curves for H1N1 Flu models

Model	Pre-tuning ROC AUC	Post-tuning ROC AUC
Logistic Regression	0.839	0.854
Random Forest	0.826	0.828
LightGBM	0.872	0.875

Table 1 Model Comparison of ROC AUC for H1N1 Flu models

After comparing multiple models, LightGBM demonstrated the strongest performance, achieving the highest ROC AUC for the H1N1 vaccine prediction task. This suggests LightGBM was most effective at ranking individuals by their likelihood of receiving the vaccine across all decision thresholds.

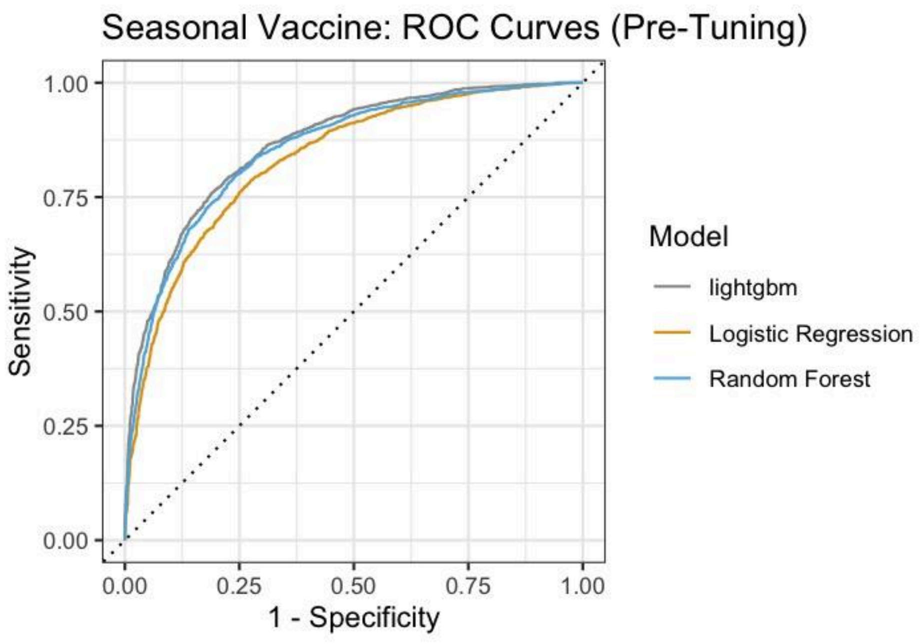

Fig. 13 Model Comparison of pre-tuning ROC curves for Seasonal Flu models

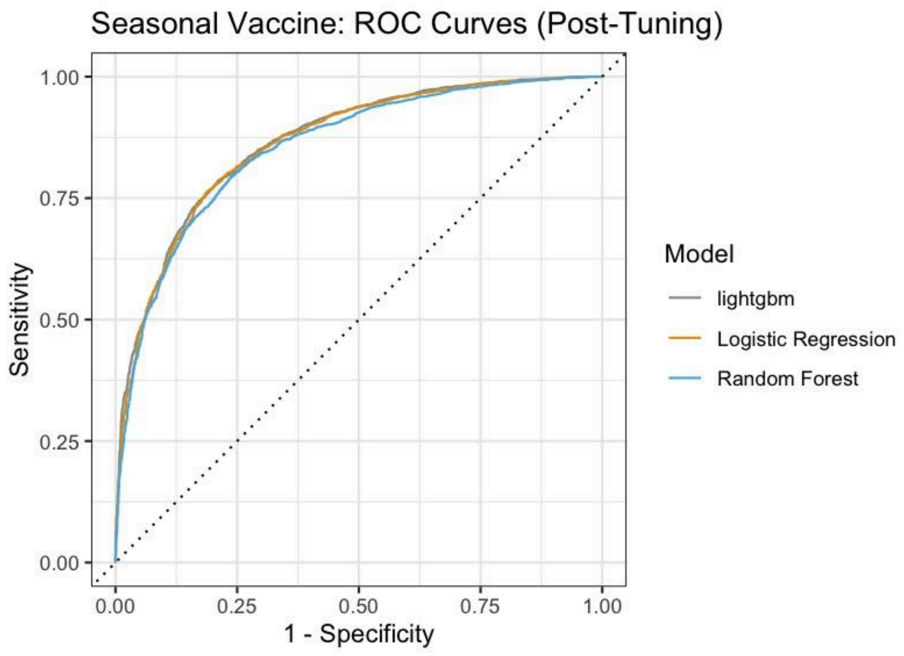

Fig. 14 Model Comparison of post-tuning ROC curves for Seasonal Flu models

Model	Pre-tuning ROC AUC	Post-tuning ROC AUC
Logistic Regression	0.846	0.859
Random Forest	0.853	0.854
LightGBM	0.865	0.867

Table 2 Model Comparison pre-tuning ROC and AUC for Seasonal Flu models

After comparing multiple models, LightGBM demonstrated the strongest performance, achieving the highest ROC AUC for the Seasonal vaccine prediction task. This suggests LightGBM was most effective at ranking individuals by their likelihood of receiving the vaccine across all decision thresholds.

If we now compare the actual AUC scores which are used to rank the models in the competition we will also be able to compare their rankings. The final AUC is determined by taking the mean of the AUC scores for the H1N1 and Seasonal Flu models.

Model	Public Leaderboard Score	Rank
Logistic Regression	0.8542	836
Random Forest	0.8392	-
LightGBM	0.8625	230

Table 3 Public ROC AUC score and ranking of each Model

After submitting the models to the competition website, we get the confirmation that the LightGBM model performs the best out of all models we fitted. It prevails with an AUC of 0.8625 placing us on the 230th rank which makes us the top 2.8% on the leaderboard out of all participants.

6. Conclusion [Vanilton Paulo]

Through experimentation with multiple approaches, we found that LightGBM outperformed all previous models, achieving an AUC value of 0.8625 in the public leaderboard. This score ranked us 230th at the time this report was written, placing us in the top 2.8% of all participants. Thus suggesting that LightGBM can effectively distinguish between vaccinated and non-vaccinated individuals.

The explorative data analysis revealed patterns in vaccination behavior. For Seasonal Flu, vaccination rates of older individuals, who face higher risks of flu-related complications, were higher. The H1N1 vaccine showed that, despite older adults being at a higher risk of complications, they were less likely to get infected, and the vaccination rate reflected this lower perceived risk.

In terms of model building, we determine that various models offer pros and cons in terms of predictive power and runtime. Simpler models, such as decision trees and bagged forests, perform below random forests in this project. Although Xgboost is a highly praised model for its predictive power, in terms of iteration, it may not be the best, with LightGBM being faster and easier to iterate.

Regarding tuning, we found that the default parameters are not always ideal. One could potentially be missing out on potential higher predictive power. However, by providing ranges as given by the mlr3tuning package, we can achieve higher scores.

This competition demonstrates how machine learning can be applied to public health decision-making, providing insights into which demographics are less likely to receive vaccines and potentially guiding targeted intervention strategies.

For future work, we recommend Model Ensembling and Stacking. By training multiple models, we can build a meta-model using, i.e. CatBoost and LightGBM and combine their predictions.

Another approach would be to use Voting/Averaging. Although simple, this can be effective by simply averaging predictions from multiple strong models.

Contributions of Authors

In the development of this seminar project, the contributions of specific authors were as follows:

Nhi Nguyen

- ➢ Set up the Git repository we used to manage our project.
- ➢ Kept logs of half our meetings.
- ➢ Conduct Explorative Data Analysis as well as investigate Missingness in the data.
- ➢ Performed data preprocessing for both Explorative Data Analysis and models.
- ➢ Conduct feature engineering for our models by building recipes.

Vanilton Paulo

- ➢ Decided on the Machine learning framework used in the project.
- ➢ Set up the CIP server used to run all experiments.
- ➢ Assign tasks to each team member.
- ➢ Kept logs of half our meetings.
- ➢ Develop the workflow used in all models from Cross-validation, Hypertuning to predicting on test dataset.
- ➢ Design the structure on the repo.
- ➢ Maintained a log of random seeds, parameters, and submissions.

List of Figures

List of Tables

AI Tools

We hereby confirm that we have written all parts of this work ourselves. For programming we have used ChatGPT-4 in the EDA section of the project, after critical review, we adopted most of the resulting suggestions for improvement.
We also used grammarly throughout the report to fix grammar and punctuation as well as improve our writing style. These suggestions were not taken verbatim, but were manually reviewed, adapted and rephrased.

References

[1,2] DrivenData, *Flu Shot Learning - Predict H1N1 and Seasonal Flu Vaccines*, https://www.drivendata.org/competitions/66/flu-shot-learning/page/213/

[3] R. Elliot, *What is Random Digit Dialing?*, GeoPoll, https://www.geopoll.com/blog/what-is-random-digit-dialing/

[4] Google Developer's, *Datasets: Imbalanced dataset*s, https://developers.google.com/machine-learning/crash-course/overfitting/imbalanced-dataset

[5] Z. Bobbitt, *Phi Coefficient: Definition and Examples, Stratology*, https://www.statology.org/phi-coefficient/

[6] A. Heiss, *Little's missing completely at random (MCAR) test*, R Documentation, https://search.r-project.org/CRAN/refmans/naniar/html/mcar_test.html

[7] M. C. M. de Goeij, M. van Diepen, K. J. Jager, G. Tripepi, C. Zoccali, F. W. Dekker, *Multiple imputation: dealing with missing data, Nephrology Dialysis Transplantation*, Volume 28, Issue 10

[8] DrivenData, *Flu Shot Learning - Predict H1N1 and Seasonal Flu Vaccines*, https://www.drivendata.org/competitions/66/flu-shot-learning/page/211/

[9] Max Kuhn and Hadley Wickham. Tidymodels: a collection of packages for modeling

and machine learning using tidyverse principles., 2020.

[10] scikit-learn, *Cross-validation: evaluating estimator performance*, https://scikit-learn.org/stable/modules/cross_validation.html

[11] F. Lee, *What is logistic regression?*, IBM, https://www.ibm.com/think/topics/logistic-regression

[12] A. Chopra, *Stratified Split: Why Data Splitting is crucial in Machine Learning*, LinkedIn, https://www.linkedin.com/pulse/ml-6-stratified-split-why-data-splitting-crucial-machine-chopra-zdv

[13] M. Kuhn, K. Johnson, *Feature Engineering and Selection: A Practical Approach for Predictive Models*, https://bookdown.org/max/FES/

[14] A. Rojo-Echeburúa, *What Is One Hot Encoding and How to Implement It in Python*, DataCamp, https://www.datacamp.com/tutorial/one-hot-encoding-python-tutorial

[15] scikit-learn, *Cross-validation: evaluating estimator performance*, https://scikit-learn.org/stable/modules/cross_validation.html

[16] scikit-learn, *Tuning the hyper-parameters of an estimator*, https://scikit-learn.org/stable/modules/grid_search.html

[17] Marc Becker. mlr3tuningspaces: Search Spaces for 'mlr3', 2025. R package version 0.6.0.

[18] scikit-learn, *confuision_matrix*, https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html

[19,20] Google Developer's, *Classification: ROC and AUC*, https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

[21] GeeksForGeeks, *Tree Based Machine Learning Algorithms*, https://www.geeksforgeeks.org/machine-learning/tree-based-machine-learning-algorithms/

[22] Wright MN, König IR, *Splitting on categorical predictors in random forests*

[23] A.Ahmad, *A Practical Guide To Learning Models Without Normalization*, https://www.tambenaconsulting.com/blog/learning-models-with-no-normalization/

[24] M. Kuhn, D. Vaughan. *parsnip: A Common API to Modeling and AnalysisFunctions*, 2025. R package version 1.3.1.

Appendix

Submission Log

For submissions all 36 features were used.

Date and time: 2025-05-22 09:53

Approach: Logistic regression baseline (grid search tuning + default parameters)

H1N1 ROC: Pre-Tuning = 0.804, Post-Tuning = 0.823
Seasonal ROC: Pre-Tuning = 0.831, Post-Tuning = 0.854
Public Leaderboard Score: 0.8349
Team Member(s): Nhi Nguyen

Date and time: 2025-06-02 11:01

Approach: Decision trees (grid search tuning + default parameters)

H1N1 ROC: Pre-Tuning = 0.691, Post-Tuning = 0.804
Seasonal ROC: Pre-Tuning = 0.757, Post-Tuning = 0.827
Public Leaderboard Score: 0.8097
Team Member(s): Vanilton Paulo

Date and time: 2025-06-02 19:51

Approach: Bagged trees (grid search tuning + default parameters)

H1N1 ROC: Pre-Tuning = 0.791, Post-Tuning = 0.822
Seasonal ROC: Pre-Tuning = 0.827, Post-Tuning = 0.848
Public Leaderboard Score: 0.8328
Team Member(s): Vanilton Paulo

Date and time: 2025-06-08 08:44

Approach: Random Forest (grid search tuning + default parameters)

H1N1 ROC: Pre-Tuning = 0.826, Post-Tuning = 0.828
Seasonal ROC: Pre-Tuning = 0.854, Post-Tuning = 0.854
Public Leaderboard Score: 0.8397
Team Member(s): Vanilton Paulo

Date and time: 2025-07-18 18:17

Approach: Random Forest (grid search tuning + mlr3 tuning parameters)

H1N1 ROC: Pre-Tuning = 0.826, Post-Tuning = 0.828
Seasonal ROC: Pre-Tuning = 0.853, Post-Tuning = 0.854
Public Leaderboard Score: 0.8393
Team Member(s): Vanilton Paulo, Nhi Nguyen

Date and time: 2025-07-20 15:49

Approach: LightGBM (changed data preparation + feature engineering (recipe) + ANOVA racing for hyperparameter tuning)

H1N1 ROC: Pre-Tuning = 0.861, Post-Tuning = 0.864
Seasonal ROC: Pre-Tuning = 0.860, Post-Tuning = 0.862
Public Leaderboard Score: 0.8623
Team Member(s): Vanilton Paulo, Nhi Nguyen

Date and time: 2025-07-23 09:38

Approach: Logistic regression baseline (grid search tuning + mlr3 tuning parameters)

H1N1 ROC: Pre-Tuning = 0.839, Post-Tuning = 0.854
Seasonal ROC: Pre-Tuning = 0.846, Post-Tuning = 0.859
Public Leaderboard Score: 0.8542
Team Member(s): Vanilton Paulo, Nhi Nguyen

Date and time: 2025-08-04 15:50

Approach: LightGBM (changed seed for splitting data)
H1N1 ROC: Pre-Tuning = 0.872, Post-Tuning = 0.875
Seasonal ROC: Pre-Tuning = 0.865, Post-Tuning = 0.867
Public Leaderboard Score: 0.8625
Team Member(s): Vanilton Paulo, Nhi Nguyen