ESKİŞEHİR TEKNİK ÜNİVERSİTESİ
ESKİŞEHİR TECHNICAL UNIVERSITY

T.C.

ESKİŞEHİR TECHNICAL UNIVERSITY

ENGINEERING FACULTY

INDUSTRIAL ENGINEERING PROJECT

PREDICTION OF H1N1 AND SEASONAL VACCINATION STATUS AND GETTING INDIVIDUALS INFORMED BY MACHINE LEARNING

ABDÜSSAMET SÖKEL

54835150270

ENM440 – ASSOC.D. GÜRKAN ÖZTÜRK

ESKİŞEHİR – 2020

Content

Dataset

1.1 Goal and about the dataset

Each row in the dataset represents one person who responded to the National 2009 H1N1 Flu Survey. The aim is to predict how likely individuals are to receive their H1N1 and seasonal flu vaccines. Specifically, the model will be predicting two probabilities: one for ‘h1n1_vaccine’ and one for ‘seasonal_vaccine’. **In a possible epidemic, people who have not been vaccinated should be informed by a SMS message.**

1.1.1 Meanings of features

For all binary variables: 0 = No; 1 = Yes.

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

---

1. h1n1_concern - Level of concern about the H1N1 flu.
    a. 0 = Not at all concerned; 1 = Not very concerned; 2 = Somewhat concerned; 3 = Very concerned.

2. h1n1_knowledge - Level of knowledge about H1N1 flu.
    a. 0 = No knowledge; 1 = A little knowledge; 2 = A lot of knowledge.

3. behavioral_antiviral_meds - Has taken antiviral medications. (binary)

4. behavioral_avoidance - Has avoided close contact with others with flu-like symptoms. (binary)

5. behavioral_face_mask - Has bought a face mask. (binary)

6. behavioral_wash_hands - Has frequently washed hands or used hand sanitizer. (binary)

7. behavioral_large_gatherings - Has reduced time at large gatherings. (binary)

8. behavioral_outside_home - Has reduced contact with people outside of own household. (binary)

9. behavioral_touch_face - Has avoided touching eyes, nose, or mouth. (binary)

10. doctor_recc_h1n1 - H1N1 flu vaccine was recommended by doctor. (binary)

11. doctor_recc_seasonal - Seasonal flu vaccine was recommended by doctor. (binary)

12. chronic_med_condition - Has any of the following chronic medical conditions: asthma or an other lung condition, diabetes, a heart condition, a kidney condition, sickle cell anemia or other anemia, a neurological or neuromuscular condition, a liver condition, or a weakened immune system caused by a chronic illness or by medicines taken for a chronic illness. (binary)

13. child_under_6_months - Has regular close contact with a child under the age of six months. (binary)

14. health_worker - Is a healthcare worker. (binary)

15. health_insurance - Has health insurance. (binary)

16. opinion_h1n1_vacc_effective - Respondent's opinion about H1N1 vaccine effectiveness.
    a. 1 = Not at all effective; 2 = Not very effective; 3 = Don't know; 4 = Somewhat effective; 5 = Very effective.

17. opinion_h1n1_risk - Respondent's opinion about risk of getting sick with H1N1 flu without vaccine.
    a. 1 = Very Low; 2 = Somewhat low; 3 = Don't know; 4 = Somewhat high; 5 = Very high.

18. opinion_h1n1_sick_from_vacc - Respondent's worry of getting sick from taking H1N1 vaccine.
    a. 1 = Not at all worried; 2 = Not very worried; 3 = Don't know; 4 = Somewhat worried; 5 = Very worried.

19. opinion_seas_vacc_effective - Respondent's opinion about seasonal flu vaccine effectiveness.
    a. 1 = Not at all effective; 2 = Not very effective; 3 = Don't know; 4 = Somewhat effective; 5 = Very effective.

20. opinion_seas_risk - Respondent's opinion about risk of getting sick with seasonal flu without vaccine.
    a. 1 = Very Low; 2 = Somewhat low; 3 = Don't know; 4 = Somewhat high; 5 = Very high.

21. opinion_seas_sick_from_vacc - Respondent's worry of getting sick from taking seasonal flu vaccine.
    a. 1 = Not at all worried; 2 = Not very worried; 3 = Don't know; 4 = Somewhat worried; 5 = Very worried.

22. age_group - Age group of respondent.

23. education - Self-reported education level.

24. race - Race of respondent.

25. sex - Sex of respondent.

26. income_poverty - Household annual income of respondent with respect to 2008 Census poverty thresholds.

27. marital_status - Marital status of respondent.

28. rent_or_own - Housing situation of respondent.

29. employment_status - Employment status of respondent.

30. census_msa - Respondent's residence within metropolitan statistical areas (MSA) as defined by the U.S. Census.

31. household_adults - Number of **other** adults in household, top-coded to 3.

32. household_children - Number of children in household, top-coded to 3.

33. phone – Phone number of observation

Data Preprocessing

2.1 Dataframe preprocessing

The training dataset as two pieces looks like below in its original form;

	respondent_id	h1n1_vaccine	seasonal_vaccine
0	0	0	0
1	1	0	1
2	2	0	0
3	3	0	1
4	4	0	0

Figure 2.1 : Dataframe including target variables

	respondent_id	h1n1_concern	h1n1_knowledge	behavioral_antiviral_meds	behavioral_avoidance	behavioral_face_mask	behavioral_wash_hands
0	0	1.0	0.0	0.0	0.0	0.0	0.0
1	1	3.0	2.0	0.0	1.0	0.0	1.0
2	2	1.0	1.0	0.0	1.0	0.0	0.0
3	3	1.0	1.0	0.0	1.0	0.0	1.0
4	4	2.0	1.0	0.0	1.0	0.0	1.0

Figure 2.2 : Dataframe including independent variables (with unseen)

To avoid a dangerous index problem while dealing with missing values, they should be combined. After that, because it will never be used, the column 'respondent_id' should be dropped.

h1n1_vaccine	seasonal_vaccine	h1n1_concern	h1n1_knowledge	behavioral_antiviral_meds	behavioral_avoidance	behavioral_face_mask	behavioral_wash_hand
0	1	3.0	2.0	0.0	1.0	0.0	1.
1	1	1.0	0.0	0.0	1.0	0.0	1.
1	1	2.0	1.0	0.0	1.0	0.0	1.
1	1	1.0	2.0	0.0	1.0	0.0	1.
0	0	1.0	1.0	0.0	1.0	0.0	0.

Figure 2.3: The final version of training dataframe (with unseen)

2.2 Data types preprocessing

Because almost all the variables are categorical, it should be better to drop the rows which include missing value. Nevertheless, We still have 13506 observation. It looks enough to train a model which includes 32 independent variables.

As can be seen although some variables originally mean 'categorical', they are saved as 'int64' or 'float64' on the dataframe.

```
Int64Index: 6437 entries, 1 to 26703
Data columns (total 34 columns):
h1n1_vaccine                   6437 non-null int64
seasonal_vaccine               6437 non-null int64
h1n1_concern                   6437 non-null float64
h1n1_knowledge                 6437 non-null float64
behavioral_antiviral_meds      6437 non-null float64
behavioral_avoidance           6437 non-null float64
behavioral_face_mask           6437 non-null float64
behavioral_wash_hands          6437 non-null float64
behavioral_large_gatherings    6437 non-null float64
behavioral_outside_home        6437 non-null float64
behavioral_touch_face          6437 non-null float64
doctor_recc_h1n1               6437 non-null float64
doctor_recc_seasonal           6437 non-null float64
chronic_med_condition          6437 non-null float64
child_under_6_months           6437 non-null float64
health_worker                  6437 non-null float64
health_insurance               6437 non-null float64
opinion_h1n1_vacc_effective    6437 non-null float64
opinion_h1n1_risk              6437 non-null float64
opinion_h1n1_sick_from_vacc    6437 non-null float64
opinion_seas_vacc_effective    6437 non-null float64
opinion_seas_risk              6437 non-null float64
opinion_seas_sick_from_vacc    6437 non-null float64
age_group                      6437 non-null object
education                      6437 non-null object
race                           6437 non-null object
sex                            6437 non-null object
income_poverty                 6437 non-null object
marital_status                 6437 non-null object
rent_or_own                    6437 non-null object
employment_status              6437 non-null object
census_msa                     6437 non-null object
household_adults               6437 non-null float64
household_children             6437 non-null float64
dtypes: float64(23), int64(2), object(9)
memory usage: 1.7+ MB
```

Figure 2.4: Primitive types of variables

The variables which mean categorical should be coordinated, and verbal categorical variables (kind of 'sex','age group','education' etc.) should be transformed to dummy and integer variables to work well.

```
Int64Index: 26707 entries, 0 to 26706
Data columns (total 43 columns):
h1n1_vaccine                                 26707 non-null int64
seasonal_vaccine                             26707 non-null int64
h1n1_concern                                 26707 non-null int64
h1n1_knowledge                               26707 non-null int64
behavioral_antiviral_meds                    26707 non-null int64
behavioral_avoidance                         26707 non-null int64
behavioral_face_mask                         26707 non-null int64
behavioral_wash_hands                        26707 non-null int64
behavioral_large_gatherings                  26707 non-null int64
behavioral_outside_home                      26707 non-null int64
behavioral_touch_face                        26707 non-null int64
doctor_recc_h1n1                             26707 non-null int64
doctor_recc_seasonal                         26707 non-null int64
chronic_med_condition                        26707 non-null int64
child_under_6_months                         26707 non-null int64
health_worker                                26707 non-null int64
health_insurance                             26707 non-null int64
opinion_h1n1_vacc_effective                  26707 non-null int64
opinion_h1n1_risk                            26707 non-null int64
opinion_h1n1_sick_from_vacc                  26707 non-null int64
opinion_seas_vacc_effective                  26707 non-null int64
opinion_seas_risk                            26707 non-null int64
opinion_seas_sick_from_vacc                  26707 non-null int64
household_adults                             26707 non-null int64
household_children                           26707 non-null int64
sex_Female                                   26707 non-null int64
age_group_18 - 34 Years                      26707 non-null int64
age_group_35 - 44 Years                      26707 non-null int64
age_group_45 - 54 Years                      26707 non-null int64
age_group_55 - 64 Years                      26707 non-null int64
education_12 Years                           26707 non-null int64
education_< 12 Years                         26707 non-null int64
education_College Graduate                   26707 non-null int64
race_Black                                   26707 non-null int64
race_Hispanic                                26707 non-null int64
race_Other or Multiple                       26707 non-null int64
income_poverty_<= $75,000, Above Poverty     26707 non-null int64
income_poverty_> $75,000                     26707 non-null int64
marital_status_Married                       26707 non-null int64
rent_or_own_Own                              26707 non-null int64
employment_status_Employed                   26707 non-null int64
census_msa_MSA, Not Principle  City          26707 non-null int64
census_msa_MSA, Principle City               26707 non-null int64
dtypes: int64(43)
memory usage: 9.0 MB
```

Figure 2.5: After transformation and get the verbal variables to dummy

After all the preprocessing operations, to make sure I have saved the last form of dataframe as ‘mukemmel.csv’*. The dataframe that the model will train on is below.

2.3 Determining training and test dataframes

By using ‘model_selection’ from Scikit-Learn, the final form of dataframe will be divided into two parts as training and testing. %30 of observations will be used for testing data. Observations will be chosen randomly.

* https://github.com/sametsoekel/flushotlearning/blob/master/mukemmel.csv

Model

3.1 Methodology

The optimal model should classify the target variable by using categorical variables. That is why CatBoost algorithm that is based on decision trees (developed to work on categorical datasets by Yandex in 2017) will be used to build a model.

If we try to explain how CatBoost algorithm works step by step;

1. In one randomly chosen row (k-th row in the training data set), we exchange one random level of this categorical feature (i-th level of x) with a number.

2. This number is usually based on the target variable (the one we want to predict) conditional on the category level. In other words, the target number is based on the expected outcome variable.

3. A splitting attribute is used to create two sets of the training data: One set that has all categories who will have greater target variable than the one computed in step 2, and the other set with smaller target variables.

We will need have 2 different models to predict two target variables (h1n1_vaccine and seasonal_vaccine). Here our target variables, I defined them as ‘y1’ and ‘y2’. Everything else is independent variables;

	h1n1_vaccine
0	0
1	0
2	1
3	0
4	1
...	...
13501	0
13502	0
13503	0
13504	0
13505	0

13506 rows × 1 columns

	seasonal_vaccine
0	0
1	1
2	1
3	0
4	1
...	...
13501	0
13502	0
13503	0
13504	0
13505	0

13506 rows × 1 columns

Figure 3.1: Target variables

3.2 Primitive Models

If we build two models in primitive without any attempt to optimize their outputs will be like;

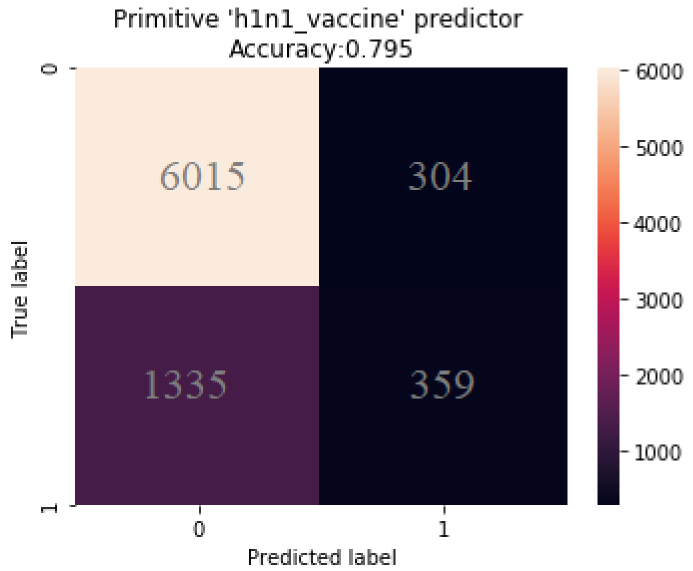

Figure 3.2: Confusion matrix of primitve 'y1' predictor.

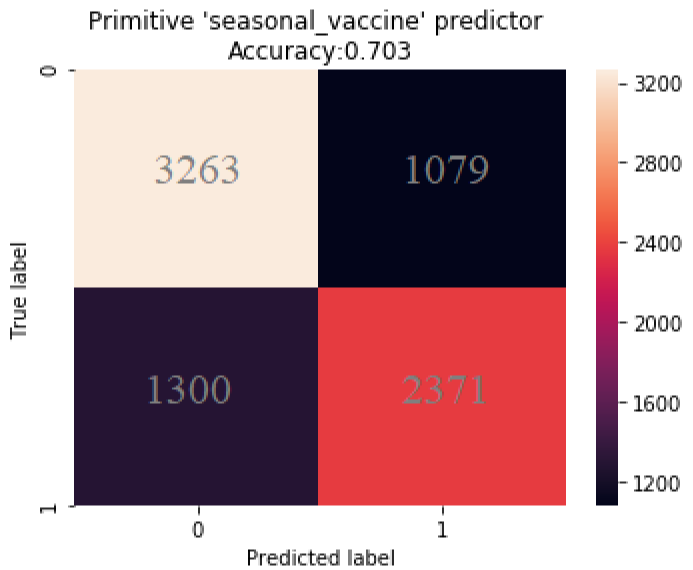

Figure 3.4: Confusion matrix of primitive 'y2' predictor.

Then according to primitive models, features importance be like respectively ;

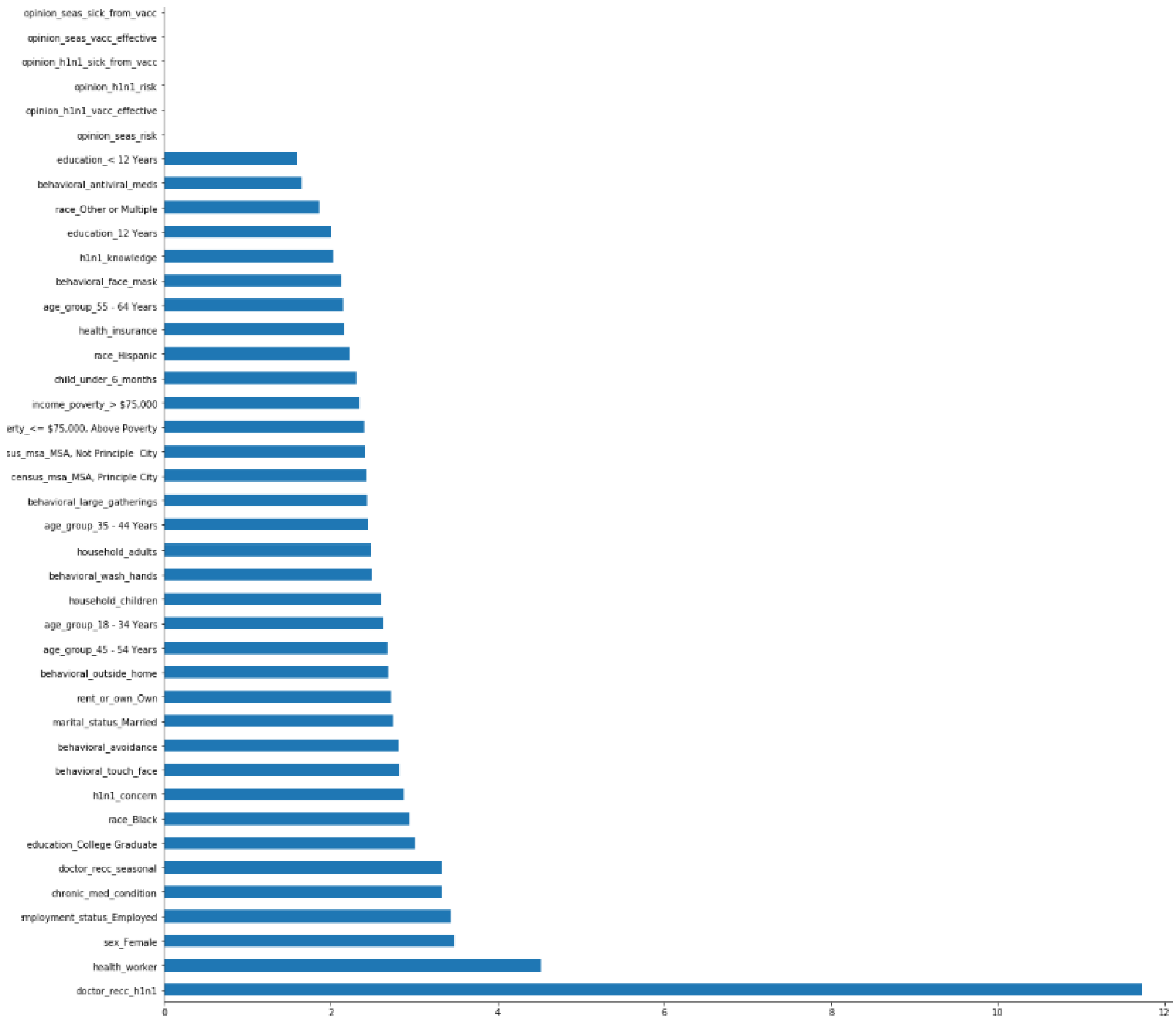

Figure 3.5: The most important feature is 'doctor_recc_h1n1'

The predictor of 'seasonal_vaccine' says almost the same information. It says the most important features is 'doctor_recc_seasonal' and 'age_group_18 – 34 Years' respectively. According to these results, **we can say that the doctor's recommendation is definitely affects people to get vaccinated or not.**

3.3 Model Tuning

GridSearchCV published by Scikit-Learn is going to help us to find the optimal hyperparameters of each model. It works on what you give as hyperparameters and how many fold of cross-validation. We can say that the library finds the optimal hyperparameters by brute-force.

I preffered 10 folds of cross-validation to validate my models well. Then to find optimal hyperparameters, I made a list of hyperparameters that are commonly used and highly recommended for CatBoost algorithm. They are like;

```
catb_params={
    'iterations':[100,200,300,500],
    'learning_rate':[0.01,0.05,0.1,0.2],
    'depth':[1,3,5,8]
}
```

Figure 3.6: Hyperparameters to be tested

I used the hyperparameters for each model, and after that different hyperparameters are determined for each.

```
{'iterations': 200, 'learning_rate': 0.05, 'depth': 8}
```

Figure 3.7: Optimal hyperparameters for 'h1n1_vaccine' predictor

```
{'depth': 3, 'iterations': 500, 'learning_rate': 0.05}
```

Figure 3.8: Optimal hyperparameters for 'seasonal_vaccine' predictor

3.4 Model Testing

After validation, the model is ready to be tested. Re-modelling required with found hyperparameters.

Their scores will be like;

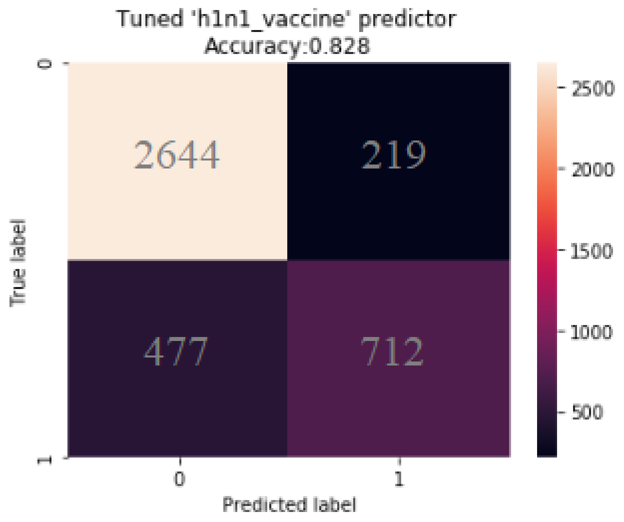

Figure 3.9: Confusion matrix of tuned 'y1' predictor

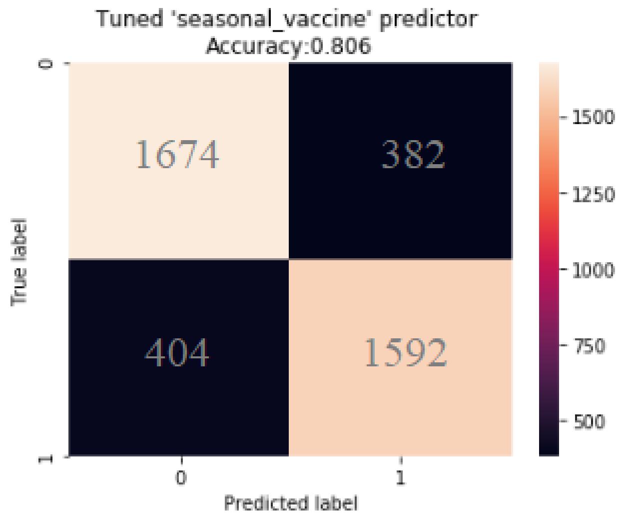

Figure 3.10: Confusion matrix of tuned 'y2' predictor

I saved both of models as ‘h1n1_predictor.cbm’ and ‘seasonal_predictor.cbm’ to use somewhere else. They can be downloaded from the link below.

https://github.com/sametsoekel/flushotlearning/blob/master/h1n1_predictor.cbm

https://github.com/sametsoekel/flushotlearning/blob/master/seasonal_predictor.cbm

Designing interfaces to visualize model

QtDesigner allows very easy and successful interface without coding. These windows are created by QtDesigner.

The first interface offers file upload as *.csv and choosing what kind of observation. Then shows contact numbers of individuals using tuned models.

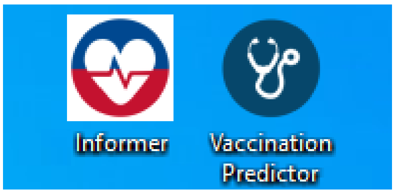

Figure 4.1: Both of interfaces

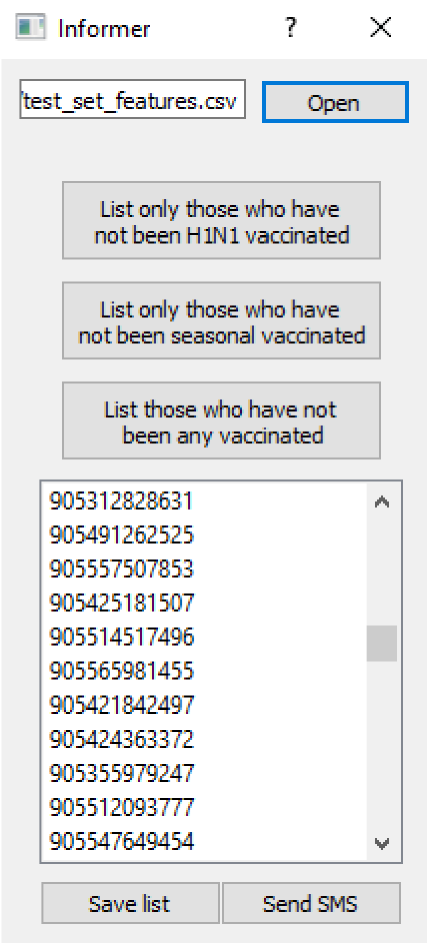

Figure 4.2: Informer interface

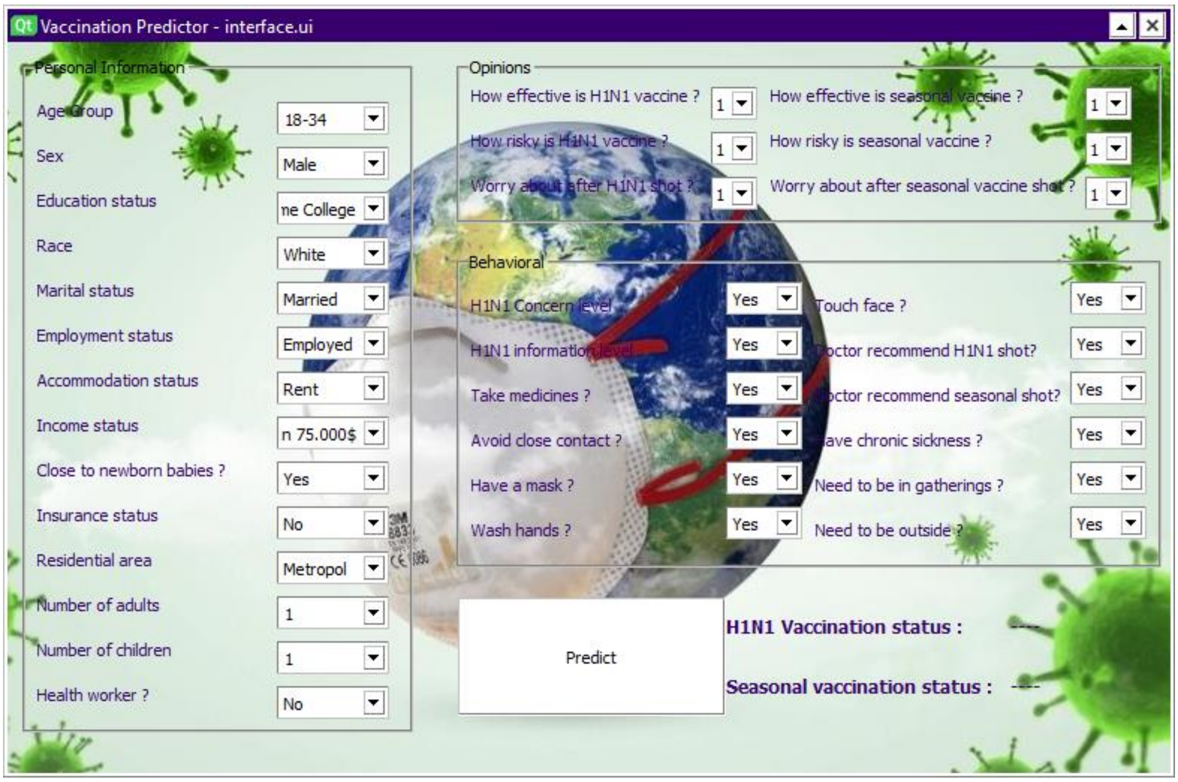

Figure 4.3: The singular prediction interface

After all arrangements the interface is ready to predict. For example;

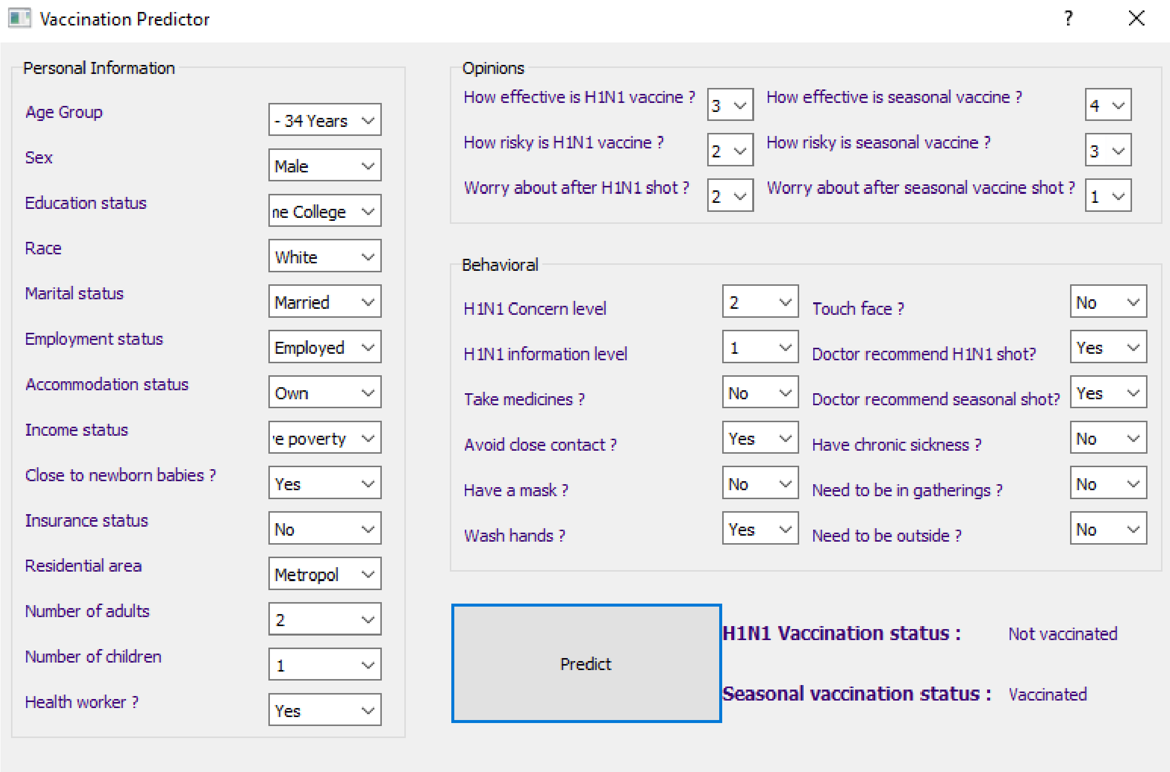

Figure 4.4: A prediction

Conclusion

After validation and optimization, both of models can make predictions with %80 accuracy. If desired, predictions can be made from dataframes or singularly.

One of the biggest advantages of CatBoost algorithm is that datasets which consisting of so many categorical variables can be interpreted very well. As because my dataset is consisting of categorical variables as well, the algorithm work with perfect accuracy.

Almost all model operations were done on JupyterNotebook, that is why file extensions might be '*.ipynb'. Data preprocessing operations are on 'work_on_dataset.**ipynb**', building tuned model and predicting operations are on 'final_models.**ipynb**' and the predictor programs are on 'program.**py**' and 'second_program.**py**'.

All the source codes, interfaces, model files are on the URL.

https://github.com/sametsoekel/flushotlearning