CFA IIT INDORE ANALYTICAX

Project Report

By DataVortex

Harshvardhan Choudhary

Kushagra Mishra

CONTENT

1. DATA CLEANING

2. DATA MANIPULATION

3. DATA VISUALIZATION

4. RESULTS AND PREDICTIONS

DATA CLEANING

The dataset provided to us was based on flu and seasonal vaccine data consisting of various fields like occupations, health insurance, race gender ,etc. We started our analysis by looking into the data for null values, there were some columns which contained a lot of null values therefore we have to drop them then we filled the rest of values using simple imputer, later we checked for data types of the columns and all the object columns were first of all converted to numerical types for giving predictions on them.

```python
#Importing all the required libraries
import pandas as pd
import numpy as np
import xgboost as xg
import matplotlib.pyplot as plt
from sklearn.impute import SimpleImputer
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import roc_auc_score
from sklearn.model_selection import RandomizedSearchCV
```

```python
#Checking for the null values
df_train_features.isnull().sum()
```

```
respondent_id                    0
h1n1_concern                    92
h1n1_knowledge                 116
behavioral_antiviral_meds       71
behavioral_avoidance           208
behavioral_face_mask            19
behavioral_wash_hands           42
behavioral_large_gatherings     87
behavioral_outside_home         82
behavioral_touch_face          128
doctor_recc_h1n1              2160
doctor_recc_seasonal          2160
chronic_med_condition          971
child_under_6_months           820
health_worker                  804
health_insurance             12274
opinion_h1n1_vacc_effective    391
opinion_h1n1_risk              388
opinion_h1n1_sick_from_vacc    395
opinion_seas_vacc_effective    462
opinion_seas_risk              514
opinion_seas_sick_from_vacc    537
age_group                        0
education                     1407
race                             0
sex                              0
income_poverty                4423
marital_status                1408
rent_or_own                   2042
employment_status             1463
hhs_geo_region                   0
```

SS for Data Cleaning

```python
#Making a list of all null columns
col = []
for i in df_train_features.columns:
  if df_train_features[i].hasnans == True:
    col.append(i)
```

```python
#Defining a common functions for cleaning data
def Manipulator(col,im):
  for i in col:
    df_train_features[i] = im.fit_transform(df_train_features[[i]])
    df_test[i] = im.transform(df_test[[i]])
```

```python
#Using Simple Impputer to fill those null values
im = SimpleImputer(strategy = 'most_frequent')
Manipulator(col,im)
```

```python
#Now checking for categorical values
col1 = []
for i in df_train_features.columns:
  if df_train_features[i].dtype == object:
    col1.append(i)
```

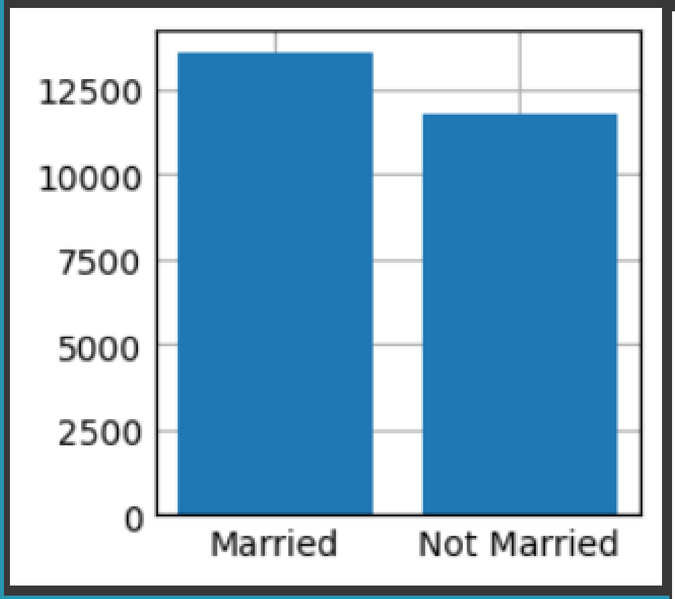
```python
#Changing categorical values to numerical values
label_encoder = preprocessing.LabelEncoder()
Manipulator(col1,label_encoder)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_label.py:116: DataC
  y = column_or_1d(y, warn=True)
```

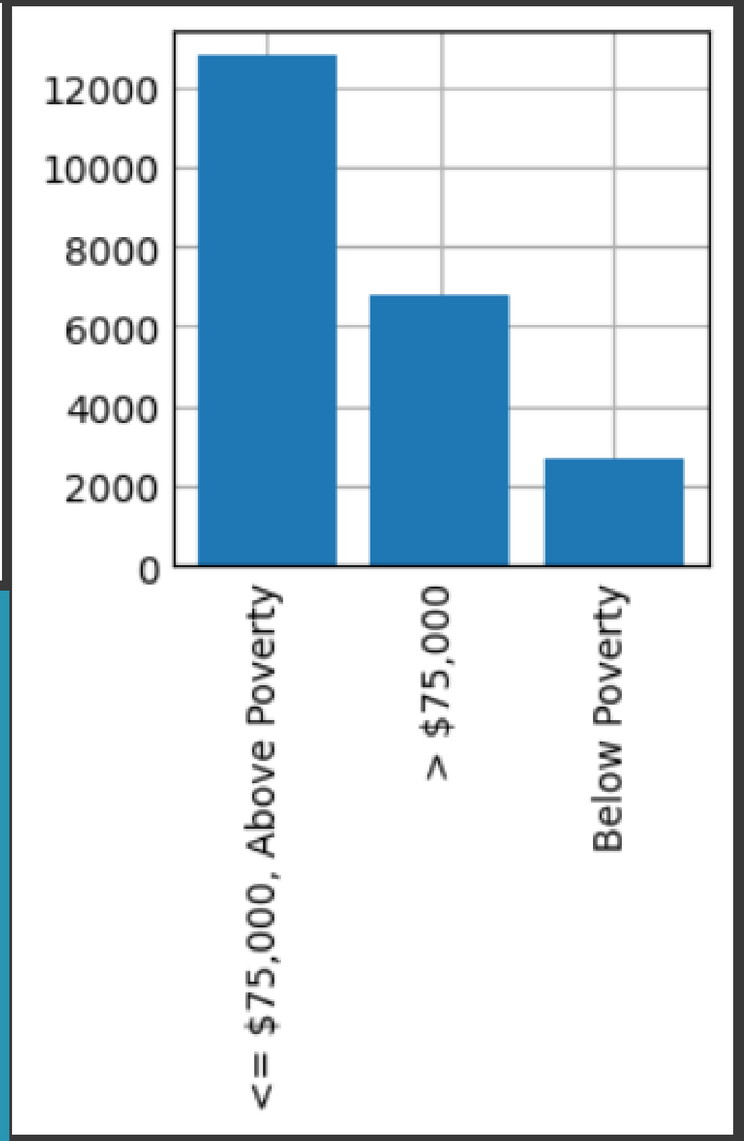
```python
#As we have a lot of null values in some of these columns we are droping it
df_train_features.drop(['employment_industry', 'employment_occupation'], axis=1, inplace=True)
df_test.drop([ 'employment_industry', 'employment_occupation'], axis=1, inplace=True)
df_train_features.columns
```

DATA MANIPULATION AND VISUALIZATION

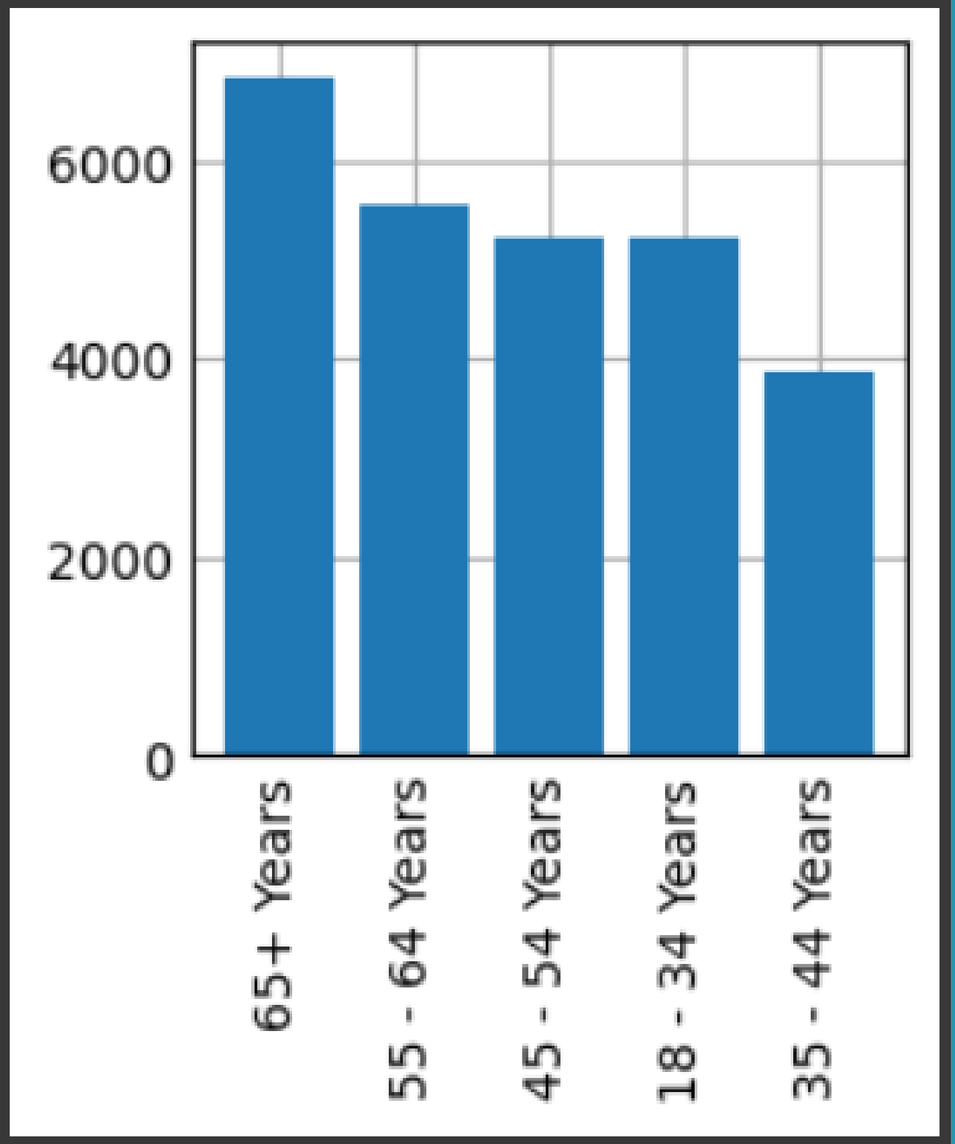
We are combining these 2 things as they are inter related in our project, we are grouping data in various ways and then looking into them as how they are giving us inferences for ex : In the plot of gender and vaccines we realzied that more female percentage took vaccines then males in both vaccine data. the following slides contain some of graphs .



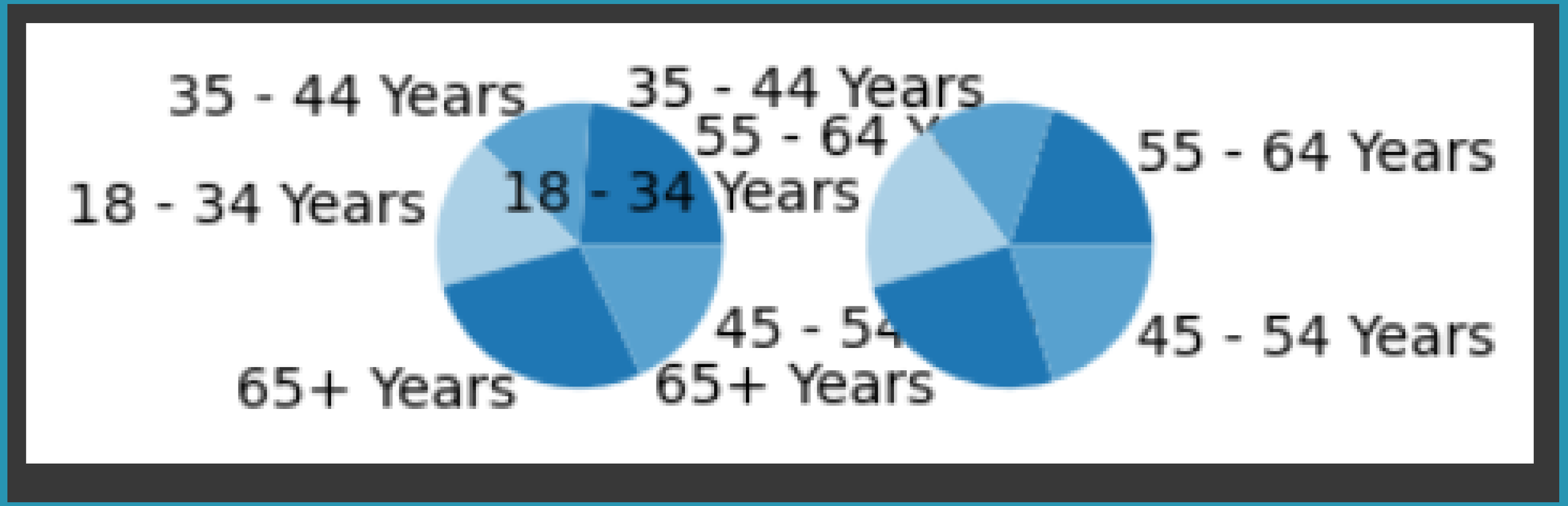
Distribution of people in dataset with respect to their marital status



Distribution of people in dataset with respect to their income group



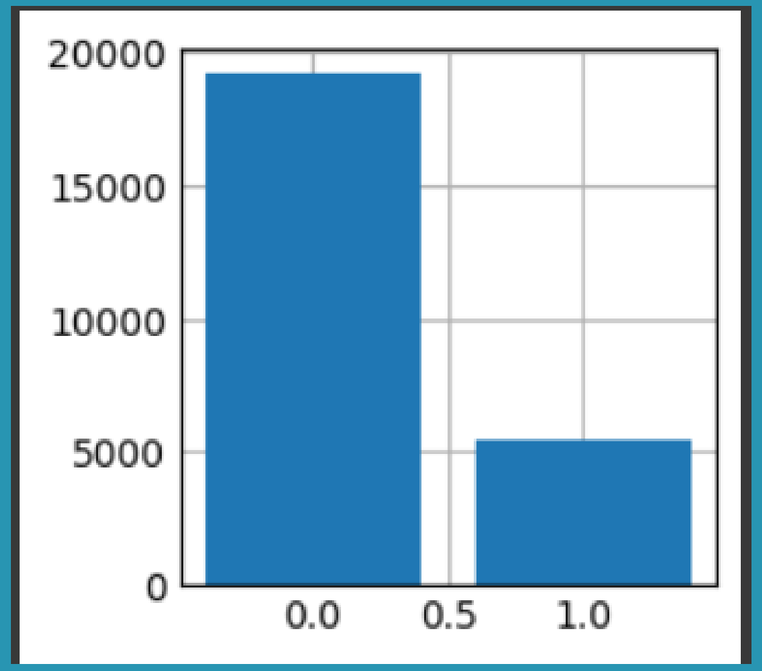
Distribution of people in dataset with respect to their age group



Representation of proportion of people who took vaccine

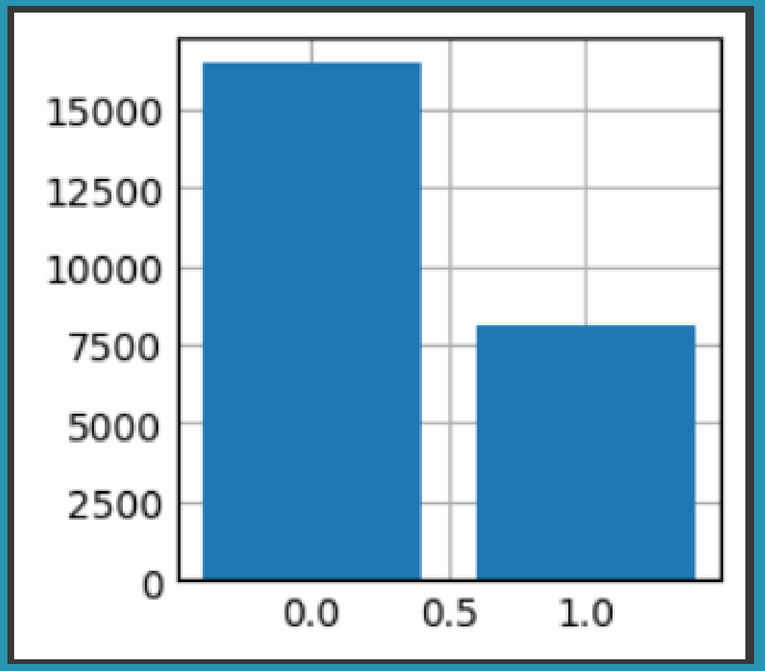
Representation of proportion of people who did not take vaccine

We concluded that age group of individual had no relation with their tendency to get vaccinated

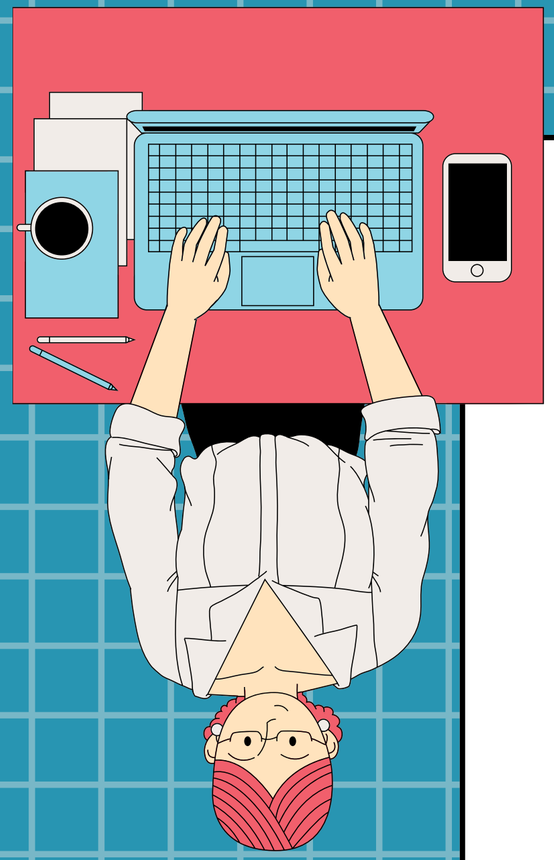


Number of people reccommended for h1n1 vaccine by doctor

1 represents yes and
0 represents no

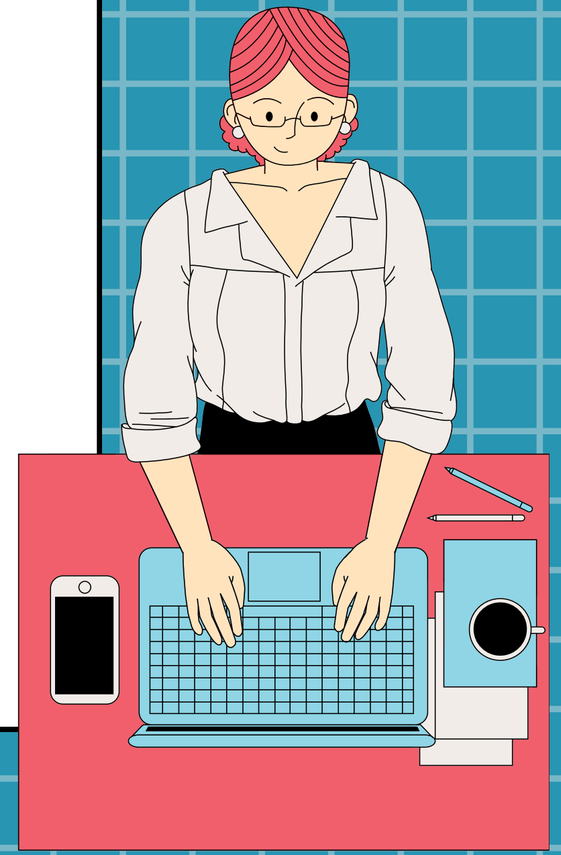


Number of people reccommended for seasonal vaccine by doctor



RESULTS AND PREDICTION

In the end of analysis we were asked to predict the probability the a person took vaccinees or not based on given test data. For the purpose we tried out 2 models RandomForestRegressor and XGBoost Regressor also we used RandomizedSearch for better results



THE PROCESS OF PREDICTION:

RESULTS AND MODEL PREDICTIONS

```
[16] X_train,X_val,y_train,y_val = train_test_split(df_train_features,df_train_labels['h1n1_vaccine'])
     model1 = xg.XGBRegressor(n_estimators = 100,seed = 100,random_state = 0)

     model1.fit(X_train,y_train)
```

```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             multi_strategy=None, n_estimators=100, n_jobs=None,
             num_parallel_tree=None, random_state=0, ...)
```

```
[26] params = {
        "n_estimators" : [50, 100, 150, 200],
      "learning_rate" : [0.05,0.10,0.15,0.20,0.25,0.30],
      "max_depth" : [ 3, 4, 5, 6, 8, 10, 12, 15],
      "min_child_weight" : [ 1, 3, 5, 7 ],
      "gamma": [ 0.0, 0.1, 0.2 , 0.3, 0.4 ],
      "colsample_bytree" : [ 0.3, 0.4, 0.5 , 0.7 ]
     }

     rs_model=RandomizedSearchCV(model1,param_distributions=params,n_iter=25,scoring='roc_auc',n_jobs=-1,cv=10,verbose=3)
     rs_model.fit(X_train,y_train)
     print(rs_model.best_estimator_)
     print(rs_model.best_score_)
```

```
Fitting 10 folds for each of 25 candidates, totalling 250 fits
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=0.3, device=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=0.1, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=0.1, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=4, max_leaves=None,
             min_child_weight=7, missing=nan, monotone_constraints=None,
             multi_strategy=None, n_estimators=200, n_jobs=None,
             num_parallel_tree=None, random_state=0, ...)
0.8670721425580108
```

26708 rows × 1 columns

```
[28] X_train,X_val,y_train,y_val = train_test_split(df_train_features,df_train_labels['seasonal_vaccine'])
     model1 = xg.XGBRegressor(n_estimators = 100,seed = 100,random_state = 0)

[29] model1.fit(X_train,y_train)
```

```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             multi_strategy=None, n_estimators=100, n_jobs=None,
             num_parallel_tree=None, random_state=0, ...)
```

```
[30] params = {
        "n_estimators" : [50, 100, 150, 200],
      "learning_rate" : [0.05,0.10,0.15,0.20,0.25,0.30],
      "max_depth" : [ 3, 4, 5, 6, 8, 10, 12, 15],
      "min_child_weight" : [ 1, 3, 5, 7 ],
      "gamma": [ 0.0, 0.1, 0.2 , 0.3, 0.4 ],
      "colsample_bytree" : [ 0.3, 0.4, 0.5 , 0.7 ]
     }

     rs_model=RandomizedSearchCV(model1,param_distributions=params,n_iter=25,scoring='roc_auc',n_jobs=-1,cv=10,verbose=3)
     rs_model.fit(X_train,y_train)
     print(rs_model.best_estimator_)
     print(rs_model.best_score_)
```

```
Fitting 10 folds for each of 25 candidates, totalling 250 fits
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=0.3, device=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=0.0, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=0.05, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=5, max_leaves=None,
             min_child_weight=7, missing=nan, monotone_constraints=None,
             multi_strategy=None, n_estimators=150, n_jobs=None,
             num_parallel_tree=None, random_state=0, ...)
0.8607029438332748
```