H1N1:

```
import numpy as np
import pandas as pd
import pandas as pd
from sklearn.model_selection import train_test_split

df = pd.merge(pd.read_csv('https://raw.githubusercontent.com/Premalatha-success/Datasets/main/h1n1_vaccine_prediction.csv')
pd.read_csv('https://raw.githubusercontent.com/Premalatha-success/Datasets/main/h1n1_vaccine_prediction.csv')
test = pd.read_csv('https://raw.githubusercontent.com/Premalatha-success/Datasets/main/h1n1_vaccine_prediction.csv ')

train, val = train_test_split(df, train_size=0.80, test_size=0.20, stratify=df[target], random_state=2)

train.shape, val.shape, test.shape
```

```
((33723, 39), (8431, 39), (28104, 38))
```

```
from pandas_profiling import ProfileReport as pr
profile = pr(train, minimal=True).to_notebook_iframe()

train.T.duplicated()
```

```
h1n1_concern                 False
h1n1_knowledge               False
behavioral_antiviral_meds    False
behavioral_avoidance         False
behavioral_face_mask         False
behavioral_wash_hands        False
behavioral_large_gatherings  False
behavioral_outside_home      False
behavioral_touch_face        False
doctor_recc_h1n1             False
doctor_recc_seasonal         False
chronic_med_condition        False
child_under_6_months         False
health_insurance             False
health_worker                False
opinion_h1n1_vacc_effective  False
opinion_h1n1_risk            False
opinion_h1n1_sick_from_vacc  False
opinion_seas_vacc_effective  False
opinion_seas_risk            False
```

```
opinion_seas_sick_from_vacc    False
agegrp                         False
education_comp                 False
raceeth4_i                     False
sex_i                          False
inc_pov                        False
marital                        False
rent_own_r                     False
employment_status              False
census_region                  False
census_msa                     False
n_adult_r                      False
household_children             False
n_people_r                     False
employment_industry            False
employment_occupation          False
hhs_region                     False
state                          False
vacc_h1n1_f                    False
dtype: bool
```

```
train.describe(exclude='number')

def engineer(df):

    behaviorals = [col for col in df.columns if 'behavioral' in col]
    df['behaviorals'] = df[behaviorals].sum(axis=1)

    fixed_data = []
    for i in df["employment_status"]:
      if i == "Not in Labor Force":
        fixed_data.append("Unemployed")
      else:
        fixed_data.append(i)
    df["employment_status"] = fixed_data

    selected_cols = df.select_dtypes(include=['number', 'object'])
    colnames = selected_cols.columns.tolist()
    labels = selected_cols.nunique()

    selected_features = labels[labels <= 30
        df = df[selected_features]

          return df


        train = engineer(train)
```

```
        val = engineer(val)
        test = engineer(test)
features = train.drop(columns=[target]).columns

X_train = train[features]
y_train = train[target]
X_val = val[features]
y_val = val[target]
X_test = test[features]
from category_encoders import OrdinalEncoder
from sklearn.impute import SimpleImputer
from sklearn.ensemble import RandomForestClassifier
from sklearn.pipeline import make_pipeline

pipe_ord = make_pipeline(
    OrdinalEncoder(),
    SimpleImputer(),
    RandomForestClassifier(n_estimators=100, random_state=10, max_depth=14, oob_score=T
rue, n_jobs=-1, criterion="gini", min_samples_split=5, max_features=6)
)

pipe_ord.fit(X_train, y_train)
print(pipe_ord.score(X_val, y_val))
```

```
0.8338275412169375
CPU times: user 7.89 s, sys: 90.7 ms, total: 7.98 s
Wall time: 3.2 s
```

```
pipe_ord.named_steps['randomforestclassifier'].oob_score_
```

```
0.823681167155947
```

```
y_pred_test = pipe_ord.predict(X_test)
y_pred_test = pd.Series(y_pred_test)
y_pred_test.value_counts()
```

```
0    24156
1     3948
dtype: int64
```

```
id = pd.Series(range(len(y_pred_test)))
y_pred_test = pd.Series(y_pred_test)
submission = pd.concat([id, y_pred_test], axis=1)
submission.rename(columns={0:"id", 1:target}, inplace=True)

print(submission.shape)
print(submission.value_counts(target))
```

```
submit=submission.to_csv("submitA.csv",index=False)
```