Final Project Submission

Please fill out:

- Student name: Dhruv Ragunathan
- Student pace: part time
- Scheduled project review date/time: 10/30/2023 3-3:45 PM
- Instructor name: Mark Barbour
- Blog post URL:[https://medium.com/p/bedb7150bbc0/ (https://medium.com/p/bedb7150bbc0/)](https://medium.com/p/bedb7150bbc0/)

Introduction and Business Case

As the world struggles to vaccinate the global population against COVID-19, an understanding of how people's backgrounds, opinions, and health behaviors are related to their personal vaccination patterns can provide guidance for future public health efforts. The stakeholders for this study are public health officials responsible for determining vaccination strategy.

The goal of this project is too predict whether people got H1N1 and seasonal flu vaccines using data collected in the National 2009 H1N1 Flu Survey. This is a binary classification problem where we will be investigating if a respondent received the Seasonal flu vaccine.

In this study, we will predict whether a participant will get the seasonal flu vaccine. We will optimize on identifying the most amount of people who need a vaccine, at the expense of erroneously identifying people who have already got the vaccine. In data science terms, we will optimize on reducing the amount of false negatives.

The rationale here is that most people who die from the flu are unvaccinated. Ensuring that people are vaccinated will save lives.

Data Understanding

- There are three sets of CSVs provided as part of this study.
- Two of the CSVs are for modeling training, and one is for model testing.
- A train - test split has already been done for us, but we will perform a train - test split on the training data anyway.
- The CSV 'training set features' provides columns we can use to target whether a participant got the Flu Vaccine or not
- The CSV training set labels contains the target column.

Steps to Get the Dataset

Here are the steps to replicate the data

- Go to https://www.kaggle.com/datasets/imdevskp/h1n1-swine-flu-2009-pandemic-dataset (https://www.kaggle.com/datasets/imdevskp/h1n1-swine-flu-2009-pandemic-dataset)
- Download data.csv
- The data is already separated into a training and testing set
- Features of the data described in training_set_labels.csv

Training Data Set

* The training data set with the features contain 26,707 entries with 35 different columns. The columns are listed out below.

Target Training Data Set

- The target feature set contains 26707 entries with 3 columns:
    - Respondent ID
    - H1N1 Vaccine Status
    - Seasonal Flu Vaccine Status

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
# Surpress warning errors

import warnings
warnings.filterwarnings('ignore')
```

In [3]:
```
X = pd.read_csv('Data/training_set_features.csv', index_col='respond
y = pd.read_csv('Data/training_set_labels.csv',index_col='respondent
```

In [4]:
```
X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 26707 entries, 0 to 26706
Data columns (total 35 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   h1n1_concern                 26615 non-null  float64
 1   h1n1_knowledge               26591 non-null  float64
 2   behavioral_antiviral_meds    26636 non-null  float64
 3   behavioral_avoidance         26499 non-null  float64
 4   behavioral_face_mask         26688 non-null  float64
 5   behavioral_wash_hands        26665 non-null  float64
 6   behavioral_large_gatherings  26620 non-null  float64
 7   behavioral_outside_home      26625 non-null  float64
 8   behavioral_touch_face        26579 non-null  float64
 9   doctor_recc_h1n1             24547 non-null  float64
 10  doctor_recc_seasonal         24547 non-null  float64
 11  chronic_med_condition        25736 non-null  float64
 12  child_under_6_months         25887 non-null  float64
 13  health_worker                25903 non-null  float64
 14  health_insurance             14433 non-null  float64
 15  opinion_h1n1_vacc_effective  26316 non-null  float64
 16  opinion_h1n1_risk            26319 non-null  float64
 17  opinion_h1n1_sick_from_vacc  26312 non-null  float64
 18  opinion_seas_vacc_effective  26245 non-null  float64
 19  opinion_seas_risk            26193 non-null  float64
 20  opinion_seas_sick_from_vacc  26170 non-null  float64
 21  age_group                    26707 non-null  object 
 22  education                    25300 non-null  object 
 23  race                         26707 non-null  object 
 24  sex                          26707 non-null  object 
 25  income_poverty               22284 non-null  object 
 26  marital_status               25299 non-null  object 
 27  rent_or_own                  24665 non-null  object 
 28  employment_status            25244 non-null  object 
 29  hhs_geo_region               26707 non-null  object 
 30  census_msa                   26707 non-null  object 
 31  household_adults             26458 non-null  float64
 32  household_children           26458 non-null  float64
 33  employment_industry          13377 non-null  object 
 34  employment_occupation        13237 non-null  object 
dtypes: float64(23), object(12)
memory usage: 7.3+ MB
```

In [5]:
```
y.head(5)
```

Out[5]:

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

In [6]:
```
y.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 26707 entries, 0 to 26706
Data columns (total 2 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   h1n1_vaccine      26707 non-null  int64
 1   seasonal_vaccine  26707 non-null  int64
dtypes: int64(2)
memory usage: 625.9 KB
```

In [7]:
```
y["h1n1_vaccine"].value_counts()
```

Out[7]:
```
0    21033
1     5674
Name: h1n1_vaccine, dtype: int64
```

In [8]:
```
y["seasonal_vaccine"].value_counts()
```

Out[8]:
```
0    14272
1    12435
Name: seasonal_vaccine, dtype: int64
```

Since there is a better mix of data for the seasonal vaccine, we are dropping the h1n1 vaccine column

In [9]:
```
y.drop(columns = ["h1n1_vaccine"],axis=0,inplace=True)
```

In [10]:
```
y
```

Out[10]:

	seasonal_vaccine
respondent_id	
0	0
1	1
2	0
3	1
4	0
...	...
26702	0
26703	0
26704	1
26705	0
26706	0

26707 rows × 1 columns

Exploratory Data Analysis

Created several slices of the data to explore multiple relationships

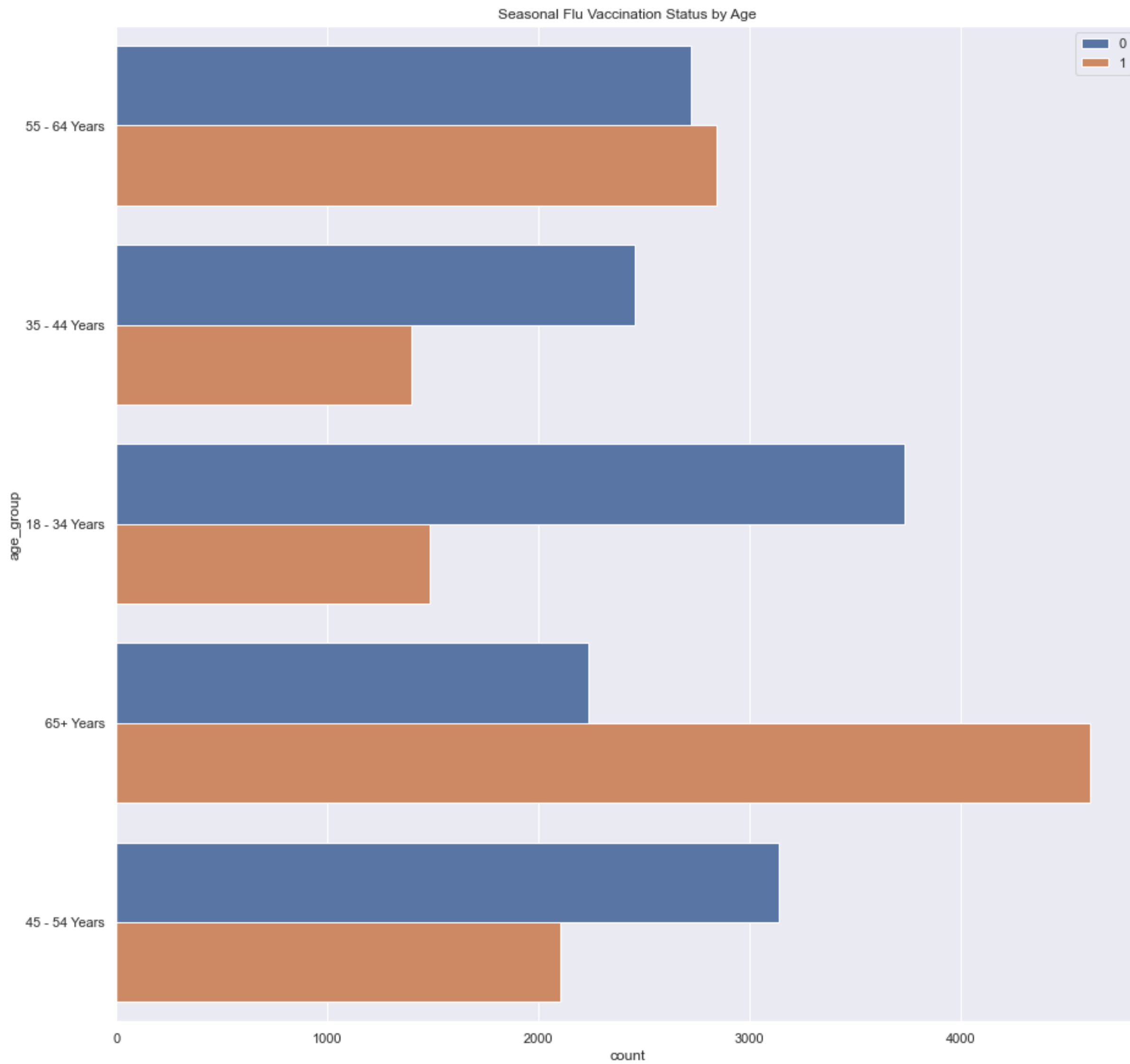
- First created a graph comparing the vaccination status of patients based on age.
  - Determined that most unvaccinated patients were in the 18 - 34 year old bucket.
- Second compared education level against vaccination status
  - Determined that more education generally correlates with a higher likelyhood of vaccination.
- Compared race to vaccination status
  - Similar disparaties off vaccination versus unvaccination between various racial groups.
- Compared gender to vaccination status
  - Females are more likely to be vaccinated than men

In [11]:
```
# Merging data columns to perform exploratory data analysis

seas_df = pd.merge(X,y,on = y.index)
```

In [12]:
```
# Compare Age against Vaccination Status
```

In [131]:
```
# Creating a histogram of the proportion of participants who were va

sns.countplot(y=seas_df['age_group'],hue=seas_df['seasonal_vaccine']
sns.set(rc={'figure.figsize':(15,15)})
plt.xticks()
plt.title('Seasonal Flu Vaccination Status by Age')
plt.legend(loc=1)
```

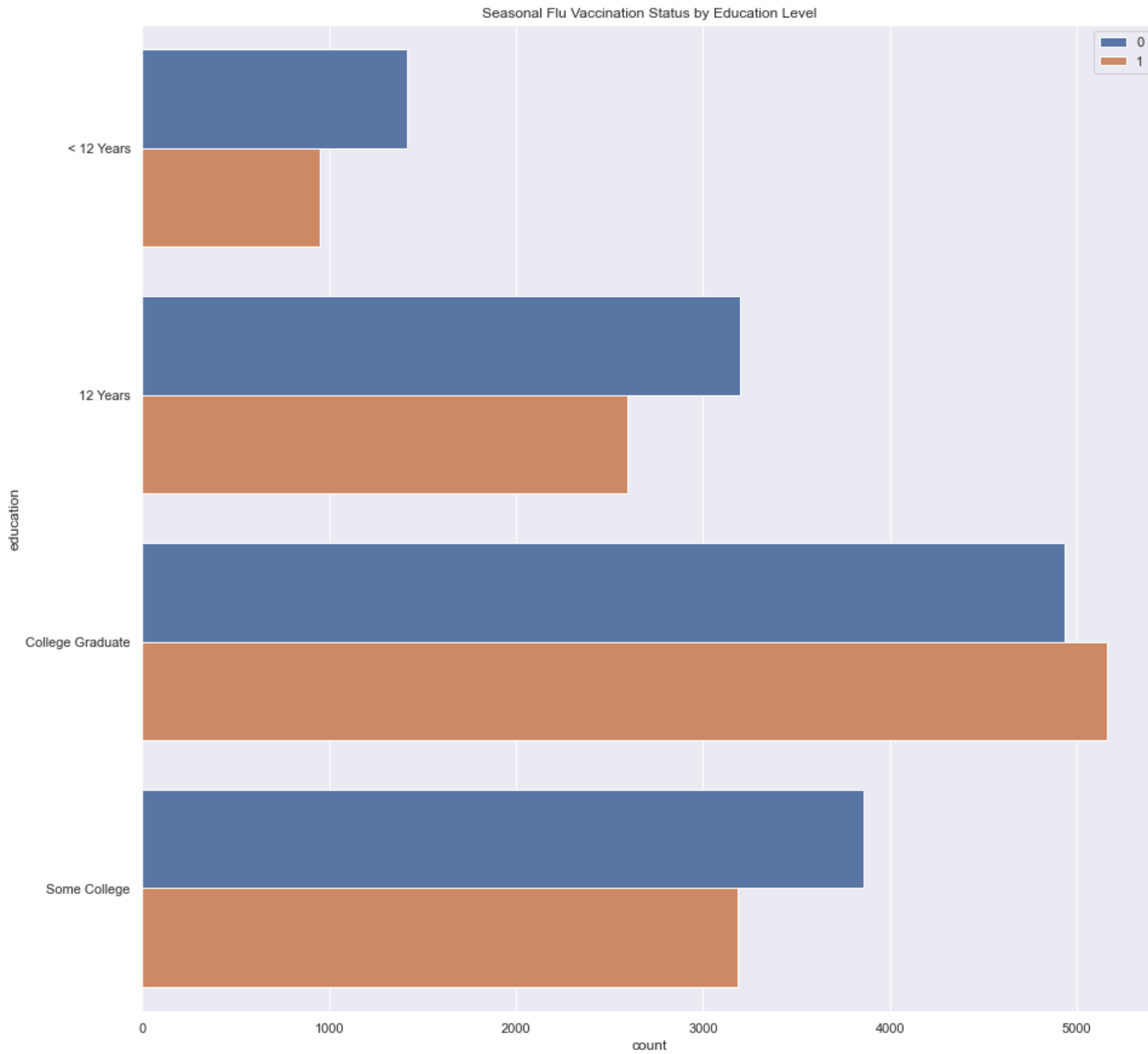
Out[131]: <matplotlib.legend.Legend at 0x7ff082fe2700>



In [14]:
```
# Compare Education Level Against Vaccination Status
```

In [15]:
```
sns.countplot(y=seas_df['education'],hue=seas_df['seasonal_vaccine']
sns.set(rc={'figure.figsize':(15,15)})
plt.xticks()
plt.title('Seasonal Flu Vaccination Status by Education Level')
plt.legend(loc=1)
```

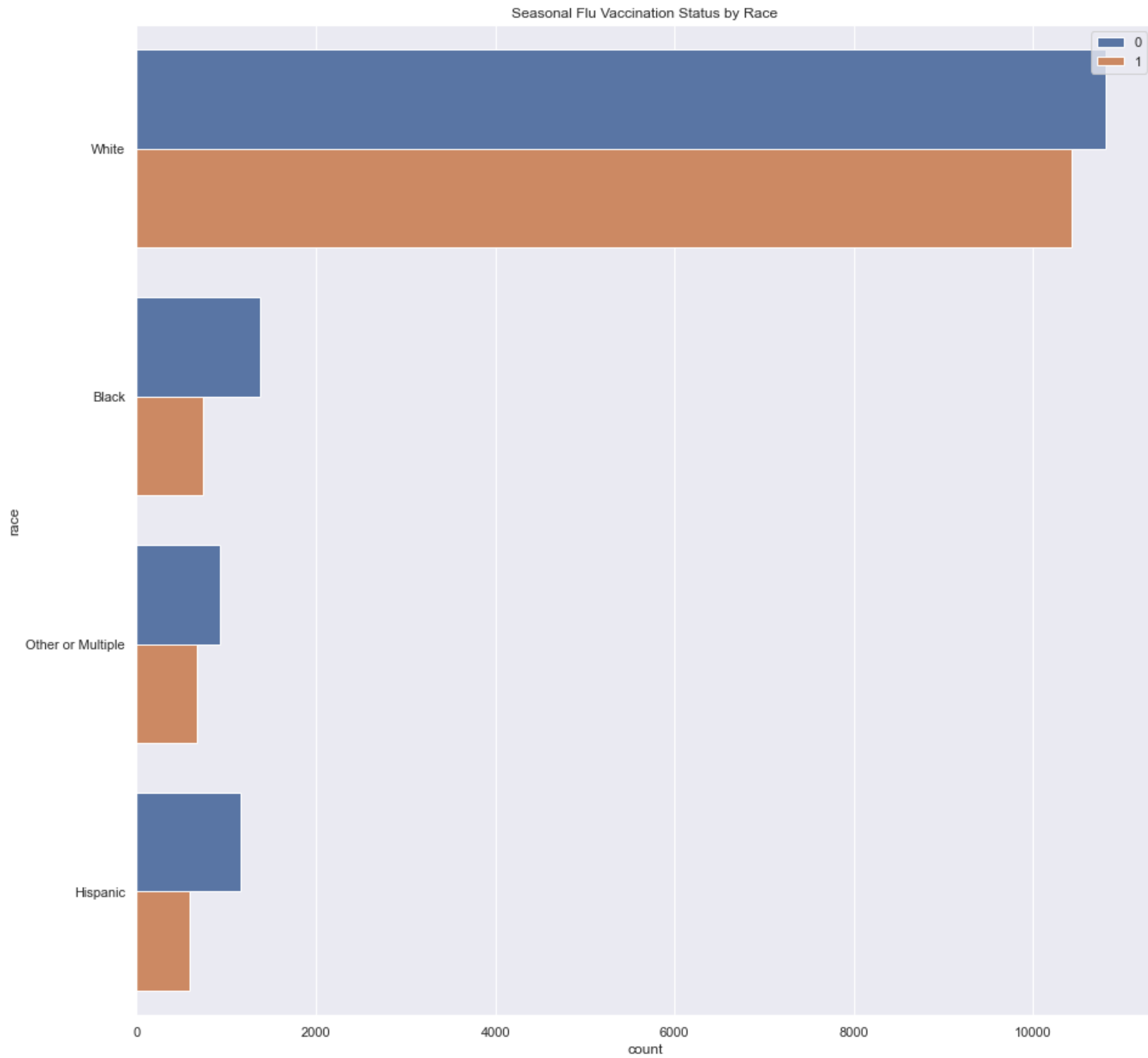
Out[15]: <matplotlib.legend.Legend at 0x7ff09760a520>



In [16]:
```
# Compare Race to Seasonal Vaccination Status
```

In [17]:
```
sns.countplot(y=seas_df['race'],hue=seas_df['seasonal_vaccine'],data
sns.set(rc={'figure.figsize':(15,15)})
plt.xticks()
plt.title('Seasonal Flu Vaccination Status by Race')
plt.legend(loc=1)
```

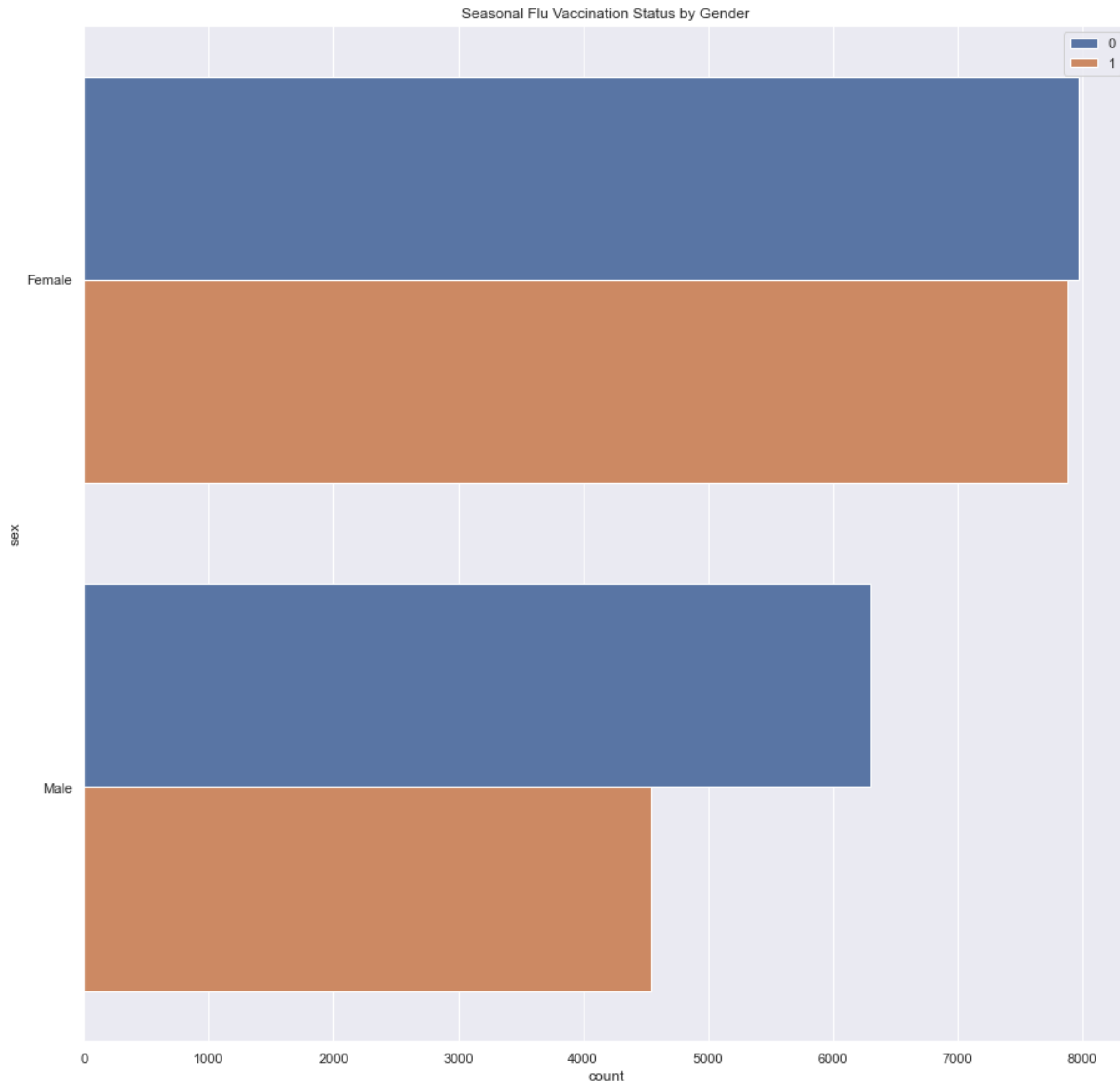
Out[17]: <matplotlib.legend.Legend at 0x7ff096ce4190>



In [18]:
```
# Compare Gender to Vaccination Status
```

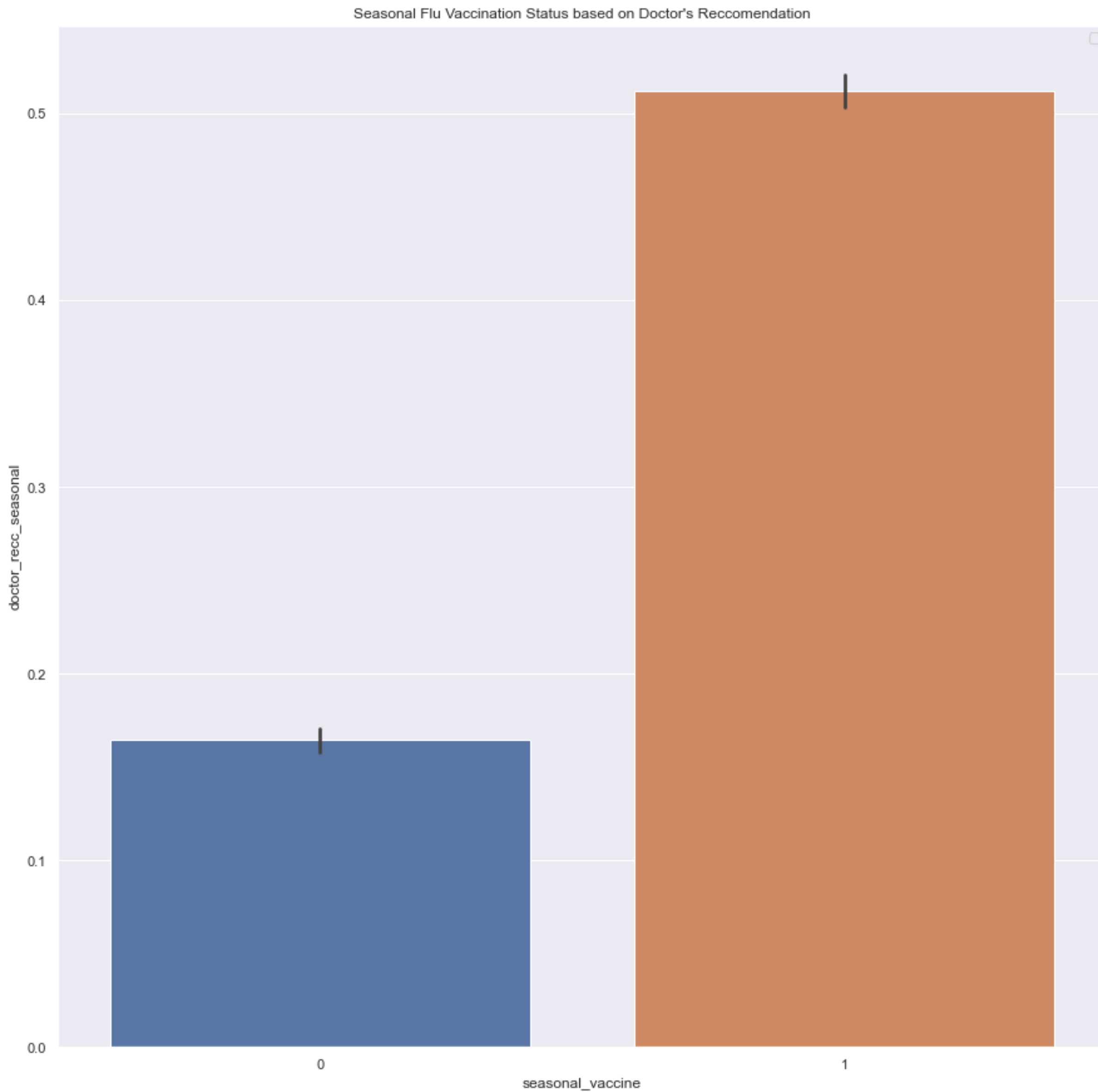
In [19]:
```
sns.countplot(y=seas_df['sex'],hue=seas_df['seasonal_vaccine'],data=
sns.set(rc={'figure.figsize':(15,15)})
plt.xticks()
plt.title('Seasonal Flu Vaccination Status by Gender')
plt.legend(loc=1)
```

Out[19]:
```
<matplotlib.legend.Legend at 0x7ff0974b93a0>
```



In [20]:
```
sns.barplot(x = seas_df['seasonal_vaccine'], y = seas_df['doctor_rec
plt.xticks()
plt.title("Seasonal Flu Vaccination Status based on Doctor's Reccome
plt.legend(loc=1)
plt.show()
```

```
No handles with labels found to put in legend.
```



Data Preparation

Here is the process we followed for data preparation

1. First dropped H1N1 columns since they are not related to the target, seasonal flu vaccination status.
2. Reviewed the missing values in each feature.
3. Columns with between 40-60% of the data missing were dropped. Those were health insurance, employment industry, employment occupation, income povery.

4. Rows were dropped where the overall missing data was under 1%: Those columns were behavioral_avoidance, behavioral_face_mask,behavioral_wash_hands, behavioral_large_gatherings, behavioral_outside_home,behavioral_touch_face,opinion_seas_vacc_effective,opinion_seas_ opinion_seas_sick_from_vacc,household_children,household_adults.
5. Rows where the amount of missing data was between 1-10% were filled with either the median for numeric or the mode for categorical data. 5a. Columns where the strategy was the fill with the median: doctor_recc_*, health_worker 5b. Columns where the strategy was too fill with the mode: chronic_med_cond, child_under_6_months, health_worker,

In [21]:
```
X.isna().sum()
```

Out[21]:
```
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
census_msa                        0
household_adults                249
household_children              249
employment_industry           13330
employment_occupation         13470
dtype: int64
```

In [22]:
```
X.columns
```

Out[22]:
```
Index(['h1n1_concern', 'h1n1_knowledge', 'behavioral_antiviral_meds',
       'behavioral_avoidance', 'behavioral_face_mask', 'behavioral_wash
_hands',
       'behavioral_large_gatherings', 'behavioral_outside_home',
       'behavioral_touch_face', 'doctor_recc_h1n1', 'doctor_recc_season
al',
       'chronic_med_condition', 'child_under_6_months', 'health_worke
r',
       'health_insurance', 'opinion_h1n1_vacc_effective', 'opinion_h1n1
_risk',
       'opinion_h1n1_sick_from_vacc', 'opinion_seas_vacc_effective',
       'opinion_seas_risk', 'opinion_seas_sick_from_vacc', 'age_group',
       'education', 'race', 'sex', 'income_poverty', 'marital_status',
       'rent_or_own', 'employment_status', 'hhs_geo_region', 'census_ms
a',
       'household_adults', 'household_children', 'employment_industry',
       'employment_occupation'],
      dtype='object')
```

In [23]:
```
#drop columns not related to Seasonal: 'h1n1_concern', 'h1n1_knowled
# Drop Columns: health insurance, employment_industry, employment oc
# drop rows: h1n1_*, behavioral_*, opinion_h1n1_vacc_effective, opin
#household_adults, household_children

# fill with median: doctor_recc_*, health_worker
# fill with mode: chronic_med_cond, child_under_6_months, health_wor



```

In [24]:
```
# List of columns dropped related to seasonal vaccine

H1N1_Columns_Dropped = ['h1n1_concern', 'h1n1_knowledge','doctor_rec

```

In [25]:
```
# List of Columns Dropped because they have large amounts of data mi

Column_Dropped_High_Data_Loss = ["health_insurance", "employment_ind
```

In [26]:
```
# List of Columns where we are dropping columns with NA

Drop_Row_Columns = ['behavioral_avoidance', 'behavioral_face_mask',
       'behavioral_large_gatherings', 'behavioral_outside_home','beh
       'opinion_seas_sick_from_vacc', 'household_children', 'househo

```

In [27]:
```
# List of columns where we fill missing values with the median and M

Fill_Median = ['health_worker']
Fill_Mode = ['chronic_med_condition', 'child_under_6_months', 'healt

```

In [28]:
```
# First drop seasonal columns

X = X.drop(H1N1_Columns_Dropped,axis = 1)

X.columns
```

Out[28]:
```
Index(['behavioral_antiviral_meds', 'behavioral_avoidance',
       'behavioral_face_mask', 'behavioral_wash_hands',
       'behavioral_large_gatherings', 'behavioral_outside_home',
       'behavioral_touch_face', 'doctor_recc_seasonal',
       'chronic_med_condition', 'child_under_6_months', 'health_worke
r',
       'health_insurance', 'opinion_seas_vacc_effective', 'opinion_seas
_risk',
       'opinion_seas_sick_from_vacc', 'age_group', 'education', 'race',
'sex',
       'income_poverty', 'marital_status', 'rent_or_own', 'employment_s
tatus',
       'hhs_geo_region', 'census_msa', 'household_adults',
       'household_children', 'employment_industry', 'employment_occupat
ion'],
      dtype='object')
```

In [29]:
```
# Drop columns that have mostly NA rows

X = X.drop(Column_Dropped_High_Data_Loss,axis = 1)

X.columns
```

Out[29]:
```
Index(['behavioral_antiviral_meds', 'behavioral_avoidance',
       'behavioral_face_mask', 'behavioral_wash_hands',
       'behavioral_large_gatherings', 'behavioral_outside_home',
       'behavioral_touch_face', 'doctor_recc_seasonal',
       'chronic_med_condition', 'child_under_6_months', 'health_worke
r',
       'opinion_seas_vacc_effective', 'opinion_seas_risk',
       'opinion_seas_sick_from_vacc', 'age_group', 'education', 'race',
'sex',
       'marital_status', 'rent_or_own', 'employment_status', 'hhs_geo_r
egion',
       'census_msa', 'household_adults', 'household_children'],
      dtype='object')
```

In [30]:
```
# Fill columns with median

for col in Fill_Median:
    X[col].fillna(X[col].median(),inplace = True)

X[Fill_Median].isna().sum()
```

Out[30]:
```
health_worker    0
dtype: int64
```

In [31]:
```
# Fill columns with Mode

for col in Fill_Mode:
    X[col].fillna(X[col].mode()[0],inplace=True)

X[Fill_Mode].isna().sum()
```

Out[31]:
```
chronic_med_condition    0
child_under_6_months     0
health_worker            0
education                0
marital_status           0
rent_or_own              0
employment_status        0
dtype: int64
```

In [32]:
```
# Finally drop rows in columns that have minimal data missing

X.dropna(axis=0,inplace=True)

X[Drop_Row_Columns].isna().sum()
```

Out[32]:
```
behavioral_avoidance            0
behavioral_face_mask            0
behavioral_wash_hands           0
behavioral_large_gatherings     0
behavioral_outside_home         0
behavioral_touch_face           0
opinion_seas_vacc_effective     0
opinion_seas_risk               0
opinion_seas_sick_from_vacc     0
household_children              0
household_adults                0
dtype: int64
```

In [33]:
```
X.isna().sum()
```

Out[33]:
```
behavioral_antiviral_meds      0
behavioral_avoidance           0
behavioral_face_mask           0
behavioral_wash_hands          0
behavioral_large_gatherings    0
behavioral_outside_home        0
behavioral_touch_face          0
doctor_recc_seasonal           0
chronic_med_condition          0
child_under_6_months           0
health_worker                  0
opinion_seas_vacc_effective    0
opinion_seas_risk              0
opinion_seas_sick_from_vacc    0
age_group                      0
education                      0
race                           0
sex                            0
marital_status                 0
rent_or_own                    0
employment_status              0
hhs_geo_region                 0
census_msa                     0
household_adults               0
household_children             0
dtype: int64
```

In [34]:
```
X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 23574 entries, 0 to 26706
Data columns (total 25 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   behavioral_antiviral_meds    23574 non-null  float64
 1   behavioral_avoidance         23574 non-null  float64
 2   behavioral_face_mask         23574 non-null  float64
 3   behavioral_wash_hands        23574 non-null  float64
 4   behavioral_large_gatherings  23574 non-null  float64
 5   behavioral_outside_home      23574 non-null  float64
 6   behavioral_touch_face        23574 non-null  float64
 7   doctor_recc_seasonal         23574 non-null  float64
 8   chronic_med_condition        23574 non-null  float64
 9   child_under_6_months         23574 non-null  float64
 10  health_worker                23574 non-null  float64
 11  opinion_seas_vacc_effective  23574 non-null  float64
 12  opinion_seas_risk            23574 non-null  float64
 13  opinion_seas_sick_from_vacc  23574 non-null  float64
 14  age_group                    23574 non-null  object 
 15  education                    23574 non-null  object 
 16  race                         23574 non-null  object 
 17  sex                          23574 non-null  object 
 18  marital_status               23574 non-null  object 
 19  rent_or_own                  23574 non-null  object 
 20  employment_status            23574 non-null  object 
 21  hhs_geo_region               23574 non-null  object 
 22  census_msa                   23574 non-null  object 
 23  household_adults             23574 non-null  float64
 24  household_children           23574 non-null  float64
dtypes: float64(16), object(9)
memory usage: 4.7+ MB
```

Data Preparation Target Dataframe

- Now let's prepare the data in y. In X we removed a lot of rows with respondant IDs that are still in our target.
- let's ensure X and y have the same respondants in the data set.
- We can do this by inner joining X and y, then only selecting for the "seasonal_vaccine" column

In [35]:

```python
# Currently there are around 26K rows

y.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 26707 entries, 0 to 26706
Data columns (total 1 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   seasonal_vaccine  26707 non-null  int64
dtypes: int64(1)
memory usage: 417.3 KB
```

In [36]:

```python
# Only leave rows that are in X

y = pd.concat(
    [y,X],
    axis=1,
    join ="inner"
)[["seasonal_vaccine"]]

y
```

Out[36]:

	seasonal_vaccine
respondent_id	
0	0
1	1
3	1
4	0
5	0
...	...
26701	0
26702	0
26703	0
26704	1
26706	0

23574 rows × 1 columns

In [37]:

```python
# y now has the same number of rows as X

assert len(y) == len(X)
```

Modeling

- First import libraries

- Train - Test Split the data
- Start with the baseline logistical model

In [38]:
```
from sklearn.model_selection import train_test_split, GridSearchCV,
from sklearn.preprocessing import StandardScaler, OneHotEncoder, Fun
from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier, GradientBoostin
from sklearn.svm import LinearSVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.multioutput import MultiOutputClassifier
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
```

In [39]:
```
## Train Test Split -

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=

```

Baseline model

- A logistical Regression model will be used as the baseline model.
- Two forms of the model were evaluated. One that used OHE and one that did not.
- Data scaled using a standard scaler after these methods.
- Train-test split occurs after transformation methods to prevent data lost.

In [40]:
```
X_train.columns
```

Out[40]:
```
Index(['behavioral_antiviral_meds', 'behavioral_avoidance',
       'behavioral_face_mask', 'behavioral_wash_hands',
       'behavioral_large_gatherings', 'behavioral_outside_home',
       'behavioral_touch_face', 'doctor_recc_seasonal',
       'chronic_med_condition', 'child_under_6_months', 'health_worke
r',
       'opinion_seas_vacc_effective', 'opinion_seas_risk',
       'opinion_seas_sick_from_vacc', 'age_group', 'education', 'race',
'sex',
       'marital_status', 'rent_or_own', 'employment_status', 'hhs_geo_r
egion',
       'census_msa', 'household_adults', 'household_children'],
      dtype='object')
```

In [41]:
```
numeric_columns = list(X_train.columns[X_train.dtypes == 'float64'].
object_columns = list(X_train.columns[X_train.dtypes == 'object'].va
```

In [42]:
```
X_train_object = X_train[object_columns]
X_test_object = X_test[object_columns]

ohe = OneHotEncoder(categories="auto", handle_unknown="ignore", spar

X_train_ohe = pd.DataFrame(ohe.fit_transform(X_train_object), column
X_test_ohe = pd.DataFrame(ohe.transform(X_test_object), columns=ohe.

X_train_ohe = pd.concat([X_train[numeric_columns], X_train_ohe], axi
X_test_ohe = pd.concat([X_test[numeric_columns], X_test_ohe], axis=1
```

In [43]:
```
# Instantiate a LogisticRegression with random_state=42
log_reg = LogisticRegression(solver="liblinear")

baseline_model = log_reg.fit(X_train_ohe, y_train)

# Make predictions on test data
y_pred = baseline_model.predict(X_test_ohe)

# Calculate the ROC-AUC score
roc_auc = roc_auc_score(y_test, y_pred)
print("ROC-AUC score: ", roc_auc)
```

```
ROC-AUC score:  0.7789120090838635
```

In [44]:
```
## Compare against model that does not use the OHE method
```

In [45]:
```
log_reg = LogisticRegression(solver="liblinear")

baseline_model = log_reg.fit(X_train[numeric_columns], y_train)

# Make predictions on test data
y_pred = baseline_model.predict(X_test[numeric_columns])

# Calculate the ROC-AUC score
roc_auc = roc_auc_score(y_test, y_pred)
print("ROC-AUC score: ", roc_auc)
```

```
ROC-AUC score:  0.756862462375863
```

In [46]:
```
## Our model did slightly better without the categorical columns sin
```

In [47]:
```
## Standardization and Scaling
```

In [48]:
```
#scaling the Dataset

ss = StandardScaler()

X_scaled = ss.fit_transform(X[numeric_columns])
X_scaled = pd.DataFrame(X,columns=numeric_columns)
X_scaled
```

Out[48]:

	behavioral_antiviral_meds	behavioral_avoidance	behavioral_face_mask	behavioral_w
respondent_id				
0	0.0	0.0	0.0	
1	0.0	1.0	0.0	
3	0.0	1.0	0.0	
4	0.0	1.0	0.0	
5	0.0	1.0	0.0	
...	...	...	...	
26701	0.0	0.0	0.0	
26702	0.0	1.0	0.0	
26703	0.0	1.0	0.0	
26704	0.0	1.0	1.0	
26706	0.0	1.0	0.0	

23574 rows × 16 columns

In [49]:
```
# Since we've lost rows from scaling the data, we need to create y-s

y_scaled = pd.concat(
    [y,X_scaled[numeric_columns]],
    axis=1,
    join ="inner"
)[["seasonal_vaccine"]]

y_scaled.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 23574 entries, 0 to 26706
Data columns (total 1 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   seasonal_vaccine  23574 non-null  int64
dtypes: int64(1)
memory usage: 368.3 KB
```

In [50]:
```
assert len(X_scaled) == len(y_scaled) #X_scaled and y_scaled are not
```

In [51]:
```
## Train Test Split for scaling

X_train_s, X_test_s, y_train_s, y_test_s = train_test_split(X_scaled
```

In [71]:
```
# Create Baseline model for Seasonal Flu Vaccine
base = baseline_model.fit(X_train_s, y_train_s)

# Make predictions on test data
y_pred2 = baseline_model.predict(X_test_s)

# Calculate the ROC-AUC score
roc_auc = roc_auc_score(y_test_s, y_pred2)
print("ROC-AUC score: ", roc_auc)
```

```
ROC-AUC score:  0.756862462375863
```

In [72]:
```
test = baseline_model.fit(X_train_s, y_train_s)
test.coef_
```

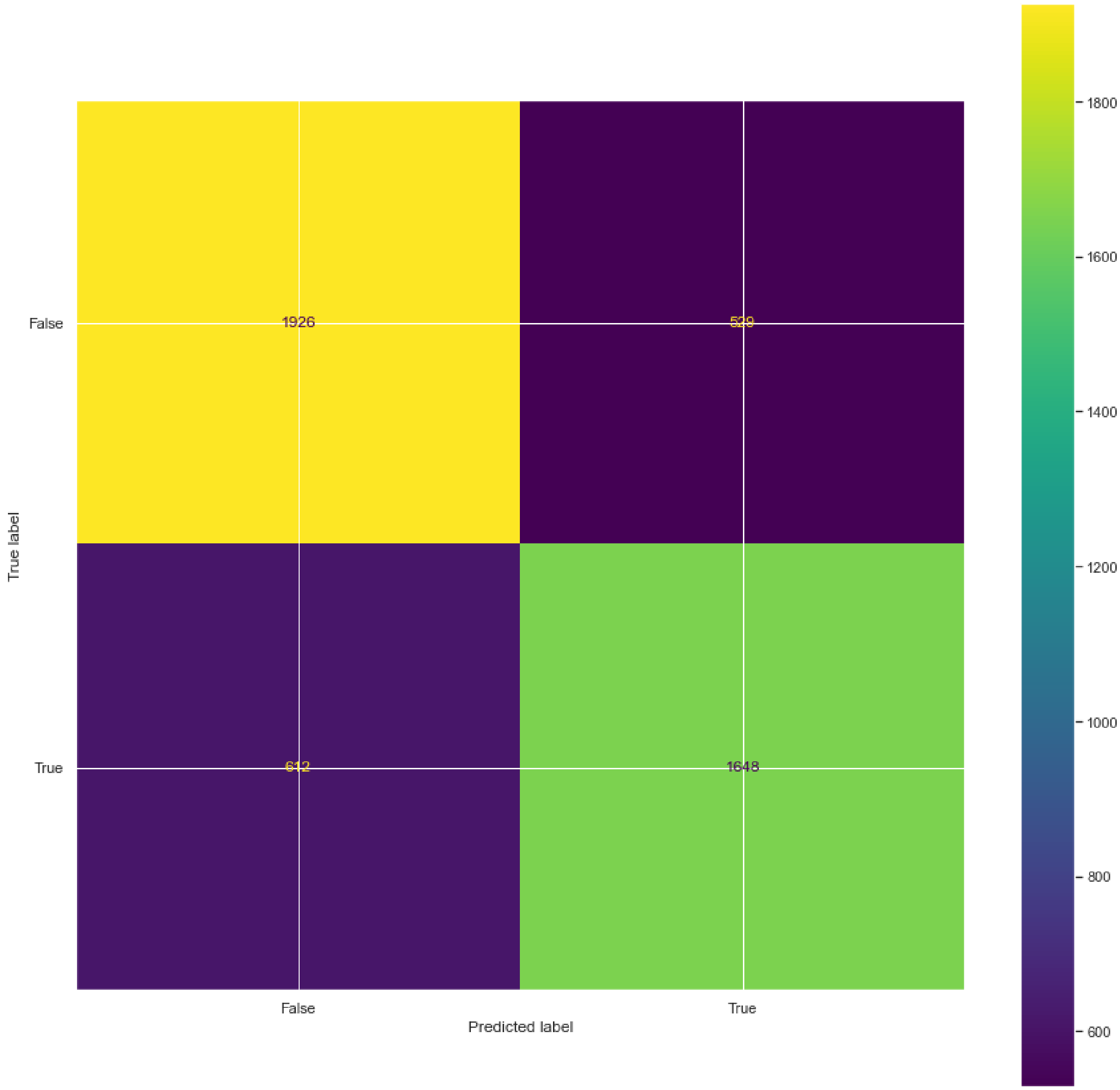
Out[72]:
```
array([[-0.21232403, -0.03735267, -0.01481719,  0.03456299, -0.0626121
6,
        -0.03795043,  0.27832577,  1.36615935,  0.3289026 , -0.0595742
3,
         0.75429765,  0.6140522 ,  0.53393888, -0.26147838, -0.1468643
,
        -0.24131809]])
```

In [54]:
```
# Scaling does not improve the ROC-AUC score.
```

In [55]:
```
# Calculate the Confusion Matrix Values

cm = confusion_matrix(y_test, y_pred2)
TN, FP, FN, TP = confusion_matrix(y_test_s, y_pred2).ravel()

print('True Positive(TP)  = ', TP)
print('False Positive(FP) = ', FP)
print('True Negative(TN)  = ', TN)
print('False Negative(FN) = ', FN)
```

```
True Positive(TP)  =  1648
False Positive(FP) =  529
True Negative(TN)  =  1926
False Negative(FN) =  612
```

In [56]:
```
from sklearn import metrics

confusion_matrix = metrics.confusion_matrix(y_test_s,y_pred2)

cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confu

cm_display.plot()
plt.show()
```



Evaluation of Baseline Model

Our baseline model has identified 1648 as true positive results. The model identified the result as positive and its true value was positive.

It had 529 False positive results. Where the model identified them as positive, but these values were actually negative.

It had 612 False Negative results. Where the model identified them as negative, but they were actually positive.

Finally, it identified 1926 true negatives. These values were negative and the model identified

Since we are trying to predict the status of someone needing a vaccine, it's preferable to reduce the number of False negatives. The use of this model would be to determine who needs a vaccination.

If the model flags a false positive, then a person who got vaccinated, would be pinged again for vaccination.

A false negative would go under the radar, potentially not receive a vaccination, which could be detrimental.

Evaluation of Other Models

- Let's evaluate other models to reduce the false negative count of the previous model
- The following models were evaluated in addition to Logistic Regression: Naive Bayes, Support Vector Machines, Decision Trees, Random Forest, and K-Nearest Neighbor.
- We calculated the following metrics to evaluate these models against one-another: accuracy, precision, recall, and Roc-AUC score.

In [57]:

```
# Initialize the other model classifiers such as logistic regression
# random forest, and K-nearest neighbors

models = {}

models['Logistic Regression'] = LogisticRegression()

models['Naive Bayes'] = GaussianNB()

models['Support Vector Machines'] = LinearSVC()

models['Decision Trees'] = DecisionTreeClassifier()

models['Random Forest'] = RandomForestClassifier()

models['K-Nearest Neighbor'] = KNeighborsClassifier()

```

In [58]:
```
# loop over each classifier to evaluate poerformance
acc, rec, prec, F1, Roc_Auc = {}, {}, {}, {}, {}

for model in models.keys():

    # Fit the classifier
    models[model].fit(X_train_s, y_train_s)

    # Make predictions
    y_pred3 = models[model].predict(X_test_s)

    # Calculate metrics
    acc[model] = accuracy_score(y_pred3, y_test_s).ravel()
    rec[model] = recall_score(y_pred3, y_test_s).ravel()
    prec[model] = precision_score(y_pred3, y_test_s).ravel()
    F1[model] = f1_score(y_pred3,y_test_s).ravel()
    Roc_Auc[model] = roc_auc_score(y_test_s, y_pred3)
```

In [59]:
```
metrics = pd.DataFrame(index=models.keys(), columns=['Accuracy','Rec
metrics['Accuracy'] = acc.values()
metrics['Recall'] = rec.values()
metrics['Precision'] = prec.values()
metrics['F1 Score'] = F1.values()
metrics['Roc-AUC Score'] = Roc_Auc.values()
metrics
```

Out[59]:

	Accuracy	Recall	Precision	F1 Scc
Logistic Regression	(0.7580063626723224,)	(0.7567691601652135,)	(0.7296460176991151,)	(0.742960126154539
Naive Bayes	(0.7257688229056204,)	(0.7117827420061322,)	(0.7190265486725663,)	(0.715386308606647
Support Vector Machines	(0.7571580063626723,)	(0.7584608252202133,)	(0.7238938053097345,)	(0.740774281186325
Decision Trees	(0.6960763520678686,)	(0.6983213429256595,)	(0.6442477876106195,)	(0.670195627157652
Random Forest	(0.7295864262990456,)	(0.7258138468592389,)	(0.7004424778761061,)	(0.712902499437063
K-Nearest Neighbor	(0.7223753976670202,)	(0.7200370198981952,)	(0.6884955752212389,)	(0.703913141823116

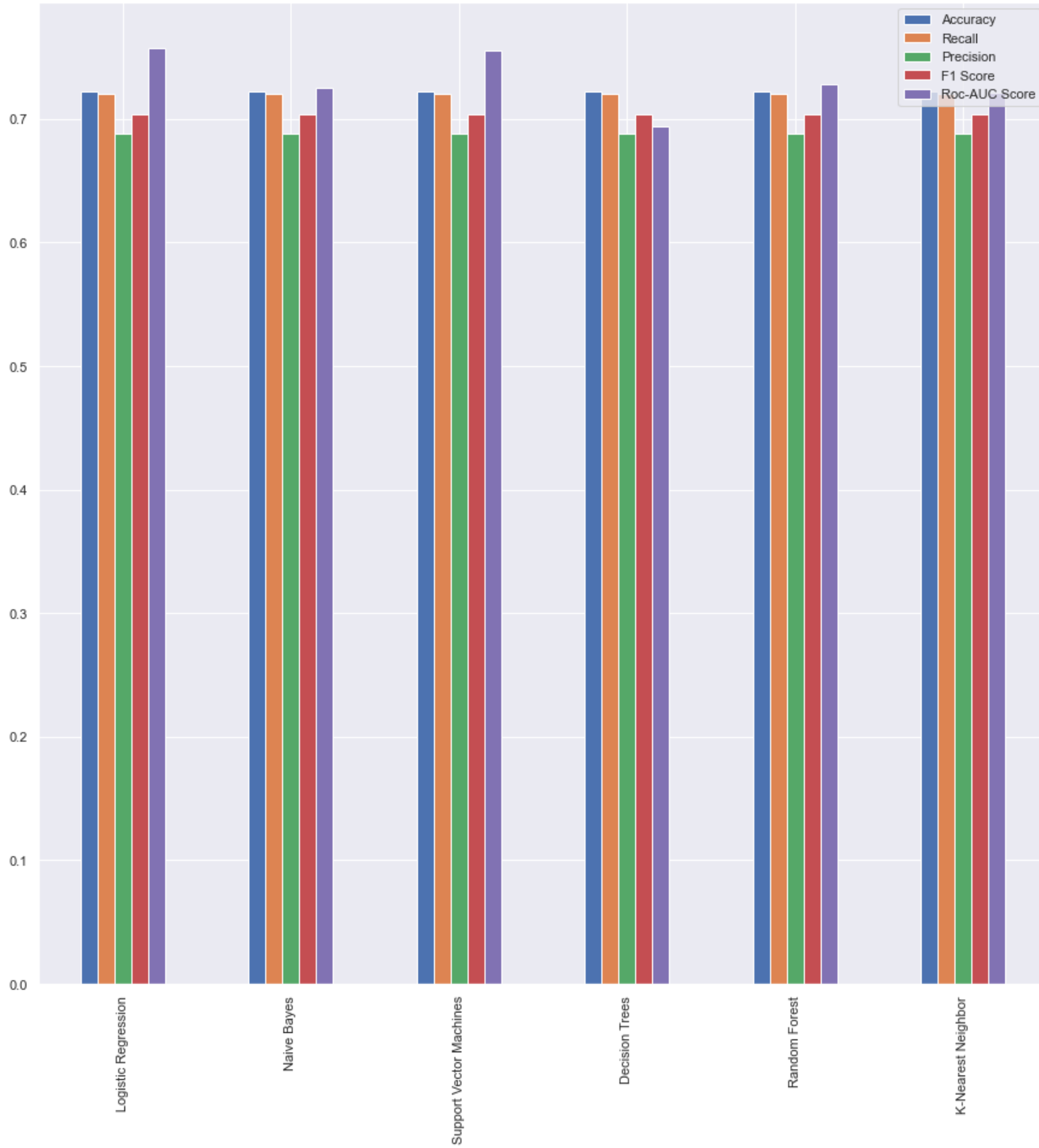
In [60]:
```
# Plot features in bar graph
# Convert objects to strings

metrics_num = metrics

for col in metrics:
    if col != 'Roc-AUC Score':
        for val in metrics[col]:
            value = val[0] #Extract the numeric value from the tuple
            metrics_num[col] = value

metrics_num
```

Out[60]:

	Accuracy	Recall	Precision	F1 Score	Roc-AUC Score
Logistic Regression	0.722375	0.720037	0.688496	0.703913	0.756880
Naive Bayes	0.722375	0.720037	0.688496	0.703913	0.725501
Support Vector Machines	0.722375	0.720037	0.688496	0.703913	0.755837
Decision Trees	0.722375	0.720037	0.688496	0.703913	0.694018
Random Forest	0.722375	0.720037	0.688496	0.703913	0.728429
K-Nearest Neighbor	0.722375	0.720037	0.688496	0.703913	0.721030

In [61]:
```
metrics_num.plot.bar()
```

Out[61]: <AxesSubplot:>



Feature Importance

In order to determine feature importance in logistical regression, the coefficients and the statistical significance are evaluated.

Coefficients that are high in magnitude and that have a high statistical significance are deemed off importance.

The features that were not statistically significant and there not included were behavioral_antiviral_meds, behavioral_large_gatherings, behavioral_outside_home, behavioral_face_mask, child_under_6_months, opionion_seas_vacc_effective.

The doctor's recommendation for seasonal flu was determined to be the most importance feature.

In [76]:
```
print("Training set score: {:.3f}".format(base.score(X_train_s, y_tr
print("Test set score: {:.3f}".format(base.score(X_test_s,y_test_s))

import statsmodels.api as sm
logit_model=sm.Logit(y_train_s,X_train_s)
result=logit_model.fit()
print(result.summary())
```

```
Training set score: 0.761
Test set score: 0.758
Optimization terminated successfully.
         Current function value: 0.559998
         Iterations 6
                           Logit Regression Results
========================================================================
=======
Dep. Variable:       seasonal_vaccine   No. Observations:
18859
Model:                          Logit   Df Residuals:
18843
Method:                           MLE   Df Model:
15
Date:                Mon, 30 Oct 2023   Pseudo R-squ.:
0.1908
Time:                        21:46:44   Log-Likelihood:
-10561.
converged:                       True   LL-Null:
-13051.
Covariance Type:            nonrobust   LLR p-value:
0.000
========================================================================
========================
                                  coef    std err          z      P>|z|
[0.025      0.975]
------------------------------------------------------------------------
------------------------
behavioral_antiviral_meds      -0.1114      0.080     -1.386      0.166
-0.269       0.046
behavioral_avoidance           -0.2255      0.042     -5.418      0.000
-0.307      -0.144
behavioral_face_mask            0.0920      0.070      1.318      0.188
-0.045       0.229
behavioral_wash_hands          -0.4848      0.048    -10.107      0.000
-0.579      -0.391
behavioral_large_gatherings    -0.0212      0.044     -0.479      0.632
-0.108       0.065
behavioral_outside_home         0.0005      0.045      0.012      0.990
-0.087       0.088
behavioral_touch_face           0.1241      0.041      3.061      0.002
0.045       0.204
doctor_recc_seasonal            1.3523      0.038     35.315      0.000
1.277       1.427
chronic_med_condition           0.1727      0.039      4.451      0.000
0.097       0.249
child_under_6_months           -0.1005      0.062     -1.614      0.106
-0.223       0.022
health_worker                   0.6257      0.056     11.248      0.000
0.517       0.735
opinion_seas_vacc_effective    -0.0082      0.012     -0.679      0.497
-0.032       0.015
opinion_seas_risk               0.5004      0.014     34.977      0.000
0.472       0.528
opinion_seas_sick_from_vacc    -0.3979      0.013    -29.537      0.000
-0.424      -0.371
household_adults               -0.3709      0.022    -16.590      0.000
```

```
-0.415      -0.327
household_children          -0.3036      0.019    -16.040      0.000
-0.341      -0.266
==============================================================================
========================
```

In [85]:
```
result.params
```

Out[85]:
```
behavioral_antiviral_meds     -0.111427
behavioral_avoidance          -0.225497
behavioral_face_mask           0.092017
behavioral_wash_hands         -0.484760
behavioral_large_gatherings   -0.021179
behavioral_outside_home        0.000538
behavioral_touch_face          0.124073
doctor_recc_seasonal           1.352343
chronic_med_condition          0.172746
child_under_6_months          -0.100540
health_worker                  0.625736
opinion_seas_vacc_effective   -0.008173
opinion_seas_risk              0.500414
opinion_seas_sick_from_vacc   -0.397889
household_adults              -0.370949
household_children            -0.303595
dtype: float64
```

In [86]:
```
result.params['doctor_recc_seasonal']
```

Out[86]:
```
1.3523425647325877
```

In [120]:
```
# Create a bar graph of the statistical significant columns and thei
# First create a dictionary of the relevant columns

ignore_cols = ['behavioral_antiviral_meds', 'behavioral_large_gather
cols = {}

for param in result.params.index:
    if param not in ignore_cols:
        cols[param] = result.params[param]

feature_series = pd.Series(cols, name = "Logistical_Regression_Featu

feature_series.sort_values()
```

Out[120]:
```
behavioral_wash_hands         -0.484760
opinion_seas_sick_from_vacc   -0.397889
household_adults              -0.370949
household_children            -0.303595
behavioral_avoidance          -0.225497
opinion_seas_vacc_effective   -0.008173
behavioral_touch_face          0.124073
chronic_med_condition          0.172746
opinion_seas_risk              0.500414
health_worker                  0.625736
doctor_recc_seasonal           1.352343
Name: Logistical_Regression_Features, dtype: float64
```

In [122]:
```
dir(feature_series.sort_values())
```

Out[122]:
```
['T',
 '_AXIS_LEN',
 '_AXIS_NAMES',
 '_AXIS_NUMBERS',
 '_AXIS_ORDERS',
 '_AXIS_REVERSED',
 '_AXIS_TO_AXIS_NUMBER',
 '_HANDLED_TYPES',
 '__abs__',
 '__add__',
 '__and__',
 '__annotations__',
 '__array__',
 '__array_priority__',
 '__array_ufunc__',
 '__array_wrap__',
 '__bool__',
 '__class__',
 '__contains__',
```

In [123]:
```
feature_series.sort_values().index
```

Out[123]:
```
Index(['behavioral_wash_hands', 'opinion_seas_sick_from_vacc',
       'household_adults', 'household_children', 'behavioral_avoidanc
e',
       'opinion_seas_vacc_effective', 'behavioral_touch_face',
       'chronic_med_condition', 'opinion_seas_risk', 'health_worker',
       'doctor_recc_seasonal'],
      dtype='object')
```

In [124]:
```
# Create a bar graph showing the coefficient magnitude

fig = plt.figure(figsize = (15, 15))
plt.bar(feature_series.sort_values().index,feature_series.sort_value

plt.xlabel("Features in Flu Dataset")
plt.ylabel("Coefficient Magnitude")
plt.title("Coefficient Magnitude in Logistical Regression Model")
plt.xticks(rotation=90)
plt.show()
```

Final Model Evaluation

All models have fairly close accuracy, recall, precision, and roc-auc scores. Based on the business problem, we would want to reduce the number of false negatives. A false negative implies that someone was not vaccinated, was predicted to not need a vaccination. That means that this person would be missed if this model would target whom to outreach.

As a result, the logistic regression model and the Support Vector machine model are the top models. Both have accuracy scores of around 76% and recall of 76%.

It seems like the baseline model using logistical regression performed better on the test data with a higher recall and accuracy score. As a result, the final model for this project will be the Logistical regression model.

The top features that influenced whether or not a person got the vaccine was the Doctor's recommendation, whether or not they were a health worker, or their opinion on the seasonal flu.

Contact Information

- Email: [Dhruvragunathan@gmail.com (mailto:Dhruvragunathan@gmail.com)](mailto:Dhruvragunathan@gmail.com)

- LinkedIn: https://www.linkedin.com/in/dhruv-ragunathan-908993b1/ (https://www.linkedin.com/in/dhruv-ragunathan-908993b1/)
- Github: https://github.com/dragunat2016/CDCH1N1 (https://github.com/dragunat2016/CDCH1N1)