Model

December 3, 2023

[1]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.metrics import accuracy_score
import pandas as pd
```

[2]:
```
# Read data from Excel file into a Pandas DataFrame
file_path = 'dm_mimic_pathways.csv'
df = pd.read_csv(file_path)
```

[3]:
```
column_name_mapping = {'person_id': 'Person',
                       'race_concept_id': 'Race',
                       'gender_concept_id':'Gender',
                       'age_group':'Age Group',
                       'pathways':'Treatment Regimen'}

race_mapping = {8527: 'White/ Hispanic',
                8516: 'Black',
                8515: 'Asian',
                0:'Unknown',
                38003592: 'Asian',
                4077359: 'Other',
                4218674: 'Unknown',
                4188159: 'White/ Hispanic',
                38003599: 'Black',
                38003574: 'Asian',
                4212311: 'Asian',
                38003600: 'Black',
                8557: 'Other',
                38003584: 'Asian',
                38003578: 'Asian',
                4087921: 'Other',
                38003615: 'Other',
                38003581: 'Asian',
                8657: 'Other',
                38003579: 'Asian',
```

```
                 38003605: 'Black',
                 38003614: 'White',
                 4213463: 'White'}

gender_mapping = {8507: 'Male',
                  8532: 'Female'}

age_mapping = {'10 - 19': 'Teens',
               '20 - 29': 'Twenties',
               '30 - 39': 'Thirties',
               '40 - 49': 'Forties',
               '50 - 59': 'Fifties',
               '60 - 69': 'Sixties',
               '70 - 79': 'Seventies',
               '80 - 89': 'Eighties',
              '> 90': 'Nineties'}
```

[4]:
```
df = df.rename(columns=column_name_mapping)
df['Race'] = df['Race'].replace(race_mapping)
df['Gender'] = df['Gender'].replace(gender_mapping)
df['Age Group'] = df['Age Group'].replace(age_mapping)
df['Age Group'].fillna('Unknown', inplace=True)
```

[5]:
```
df = df[(df['Age Group'] != 'Unknown') & (df['Race'] != 'Unknown')]
```

[6]:
```
print(len(df))
n = 4
values_to_preserve = df['Treatment Regimen'].value_counts().head(n)
print(values_to_preserve)
```

```
1746
Treatment Regimen
19071700             463
19071700,40166274    197
40164929              73
40164930              62
Name: count, dtype: int64
```

[7]:
```
def preserve_or_change(value, value_set, replacement_value):
    return value if value in value_set else replacement_value
```

[8]:
```
df['Treatment Regimen'] = df['Treatment Regimen'].apply(lambda x:␣
 ↪preserve_or_change(x, values_to_preserve, 'Other'))
df.head(5)
len(df['Treatment Regimen'].unique())
```

[8]: 5

[9]:
```python
X = df[['Age Group', 'Race', 'Gender']]
y = df['Treatment Regimen']
```

[10]:
```python
preprocessor = ColumnTransformer(
    transformers=[
            ('cat', OneHotEncoder(), ['Age Group', 'Race', 'Gender'])
        ],
        remainder='passthrough'

)
pipeline = Pipeline([
    ('preprocessor', preprocessor),
    ('classifier', LogisticRegression(multi_class='multinomial', class_weight =␣
↪'balanced'))
])
```

[11]:
```python
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

[12]:
```python
# Train the model
pipeline.fit(X_train, y_train)
```

```
/home/thecedarprince/Programs/Miniconda3/envs/datsci/lib/python3.10/site-
packages/sklearn/linear_model/_logistic.py:460: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

[12]:
```
Pipeline(steps=[('preprocessor',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('cat', OneHotEncoder(),
                                                  ['Age Group', 'Race',
                                                   'Gender'])])),
                ('classifier',
                 LogisticRegression(class_weight='balanced',
                                    multi_class='multinomial'))])
```

[13]:
```python
# Make predictions on the test set
y_pred = pipeline.predict(X_test)

# Evaluate the accuracy
```

```
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')

# Create a DataFrame with actual and predicted values
df_predictions = pd.DataFrame({
    'Actual': y_test,
    'Predicted': y_pred
})

print("Actual vs Predicted:")
print(df_predictions)
```

```
Accuracy: 0.19
Actual vs Predicted:
                 Actual Predicted
408            19071700     Other
387            19071700  40164929
803            19071700  19071700
81                Other  40164929
942            19071700  40164930
…                   …         …
596   19071700,40166274  19071700
1710              Other     Other
894               Other  40164930
1226              Other     Other
1466              Other  40164929

[350 rows x 2 columns]
```

[14]:
```
# Access the one-hot encoder from the pipeline
encoder = pipeline.named_steps['preprocessor'].named_transformers_['cat']

# Get feature names after one-hot encoding
feature_names_after_encoding = list(encoder.get_feature_names_out(X.
 ↪select_dtypes(include=['object']).columns))

# Concatenate feature names with numeric features
all_feature_names = X.select_dtypes(include=['number']).columns.tolist() +␣
 ↪feature_names_after_encoding

# Access the model from the pipeline
model = pipeline.named_steps['classifier']

# Get coefficients
coefficients = model.coef_

# Display coefficients in a DataFrame
```

```python
df_coefficients = pd.DataFrame(coefficients, columns=all_feature_names)
df_coefficients['Intercept'] = model.intercept_
df_coefficients['Class'] = model.classes_
df_coefficients.set_index('Class', inplace=True)

print("Coefficients:")
print(df_coefficients)
```

```
Coefficients:
                   Age Group_< 90  Age Group_Eighties  Age Group_Fifties  \
Class
19071700                -0.900451           -0.223234           0.208907
19071700,40166274       -0.800524           -0.761609          -0.371342
40164929                 0.377856            0.391264          -0.140900
40164930                 0.660105            0.477624           0.311239
Other                    0.663015            0.115954          -0.007904

                   Age Group_Forties  Age Group_Seventies  Age Group_Sixties  \
Class
19071700                    0.298267            -0.249473          -0.169401
19071700,40166274           0.404427            -0.482911          -0.451159
40164929                    0.078585             0.230061           0.353148
40164930                   -0.801792             0.065400           0.249865
Other                       0.020513             0.436923           0.017548

                   Age Group_Teens  Age Group_Thirties  Age Group_Twenties  \
Class
19071700                 -0.250476            0.285383            1.001611
19071700,40166274         0.737814            0.617852            1.109517
40164929                 -0.238159           -0.156240           -0.894454
40164930                 -0.085513           -0.046314           -0.833684
Other                    -0.163666           -0.700681           -0.382990

                   Race_Asian  Race_Black  Race_Other  Race_White  \
Class
19071700             0.326118    0.099418   -0.413886    0.097915
19071700,40166274   -0.754830    0.480733    0.104514    0.023622
40164929             0.112284   -0.006416   -0.093591    0.406186
40164930             0.158482   -0.550936    0.855684   -0.721212
Other                0.157947   -0.022799   -0.452721    0.193490

                   Race_White/ Hispanic  Gender_Female  Gender_Male  Intercept
Class
19071700                      -0.108432       0.003382    -0.002250   0.191038
19071700,40166274              0.148026      -0.017595     0.019659   0.196949
40164929                      -0.417302       0.060008    -0.058847   0.177910
40164930                       0.254912      -0.019981     0.016910  -0.387321
Other                          0.122797      -0.025814     0.024527  -0.178576
```