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 = 10
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
40166274              61
19071700,40164929     47
19077638              45
19030580              24
19077682              19
1513912               15
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]: 11

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

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

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

[11]:
```
# 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]:
```
# Train the model
pipeline.fit(X_train, y_train)
```

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

[13]:
```
# 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
```

```python
df_predictions = pd.DataFrame({
    'Actual': y_test,
    'Predicted': y_pred
})

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

```
Accuracy: 0.03
Actual vs Predicted:
                 Actual          Predicted
408            19071700           19030580
387            19071700           40164929
803            19071700            1513912
81                Other            1513912
942            19071700  19071700,40164929
…                     …                  …
596   19071700,40166274            1513912
1710              Other           19030580
894   19071700,40164929           19030580
1226              Other           19077682
1466              Other            1513912

[350 rows x 2 columns]
```

[14]:
```python
# 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
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)
```

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

```
Coefficients:
                   Age Group_< 90  Age Group_Eighties  Age Group_Fifties  \
Class
1513912                 -0.804046            0.551879           1.109031
19030580                 1.109500            0.921363          -0.620396
19071700                -0.955300           -0.438040           0.188859
19071700,40164929       -0.702217            0.142687          -1.034952
19071700,40166274       -0.927444           -0.899923          -0.323108
19077638                 0.484172            0.931322           0.331578
19077682                 1.194929           -0.412138           0.270031
40164929                -0.083151            0.150125          -0.139468
40164930                 0.126921            0.226781           0.250767
40166274                 0.426773           -1.154048          -0.111370
Other                    0.129863           -0.020009           0.079028

                   Age Group_Forties  Age Group_Seventies  Age Group_Sixties  \
Class
1513912                     1.571769            -2.086183           0.700448
19030580                   -1.177050             0.957491          -0.130967
19071700                    0.393527            -0.358766          -0.319092
19071700,40164929           0.652708             0.811902           0.916341
19071700,40166274           0.536845            -0.539429          -0.554161
19077638                   -0.231583             0.150010          -0.519589
19077682                   -1.214608             0.993934           0.083522
40164929                    0.175070             0.119923           0.193356
40164930                   -0.546864            -0.038925           0.143415
40166274                   -0.470312            -0.490682          -0.512664
Other                       0.310498             0.480724          -0.000608

                   Age Group_Teens  Age Group_Thirties  Age Group_Twenties  \
Class
1513912                  -0.058636           -0.577002           -0.408733
19030580                 -0.032923           -0.592680           -0.436437
19071700                 -0.089438            0.632414            0.945898
19071700,40164929        -0.013213           -0.486180           -0.286694
19071700,40166274         0.558910            1.028335            1.120487
19077638                 -0.023622           -0.688773           -0.432092
19077682                 -0.016722           -0.560984           -0.340623
40164929                 -0.080623            0.195193           -0.530238
40164930                 -0.029072            0.343439           -0.474782
40166274                 -0.168843            1.170357            1.311965
Other                    -0.045817           -0.464120           -0.468751

                   Race_Asian  Race_Black  Race_Other  Race_White  \
```

Class				
1513912	-0.538837	0.964780	-0.653399	-0.300968
19030580	-0.467859	0.392375	-0.537712	-0.278832
19071700	0.551472	-0.116899	0.114966	-0.061116
19071700,40164929	-0.534197	-0.117667	-0.675926	0.996918
19071700,40166274	-0.367821	0.149241	0.602590	-0.115440
19077638	-0.853944	-0.845337	0.827204	1.134813
19077682	-0.330971	0.390573	-0.544306	-0.363716
40164929	0.388481	-0.241259	0.467045	0.172503
40164930	0.280223	-0.750706	1.256302	-0.572986
40166274	1.414858	0.429266	-1.040592	-0.515329
Other	0.458596	-0.254368	0.183828	-0.095847

	Race_White/ Hispanic	Gender_Female	Gender_Male	Intercept
Class				
1513912	0.526952	0.176377	-0.177848	-0.896512
19030580	0.889929	0.255082	-0.257181	-1.007399
19071700	-0.488360	0.042182	-0.042119	0.776432
19071700,40164929	0.331253	-0.346342	0.346723	-0.826698
19071700,40166274	-0.268057	0.016649	-0.016137	0.775765
19077638	-0.261314	-0.102138	0.103560	0.251790
19077682	0.845761	-0.334929	0.332270	-1.109544
40164929	-0.786584	0.080891	-0.080704	0.749463
40164930	-0.211152	0.004755	-0.003074	0.260994
40166274	-0.287026	0.249647	-0.248471	0.742028
Other	-0.291402	-0.042173	0.042980	0.283679