Model

December 3, 2023

[1]:
```python
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]:
```python
# Read data from Excel file into a Pandas DataFrame
file_path = 'dm_mimic_pathways.csv'
df = pd.read_csv(file_path)
```

[3]:
```python
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 = 5
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
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]: 6

[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)
```

[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: 0.18
Actual vs Predicted:
                  Actual  Predicted
408             19071700      Other
387             19071700   40164929
803             19071700   19071700
81                 Other   40164929
942             19071700   40164930
…                    …          …
596    19071700,40166274   40166274
1710               Other      Other
894                Other   40164930
1226               Other      Other
1466               Other   40166274

[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)

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

```
Coefficients:
                   Age Group_< 90  Age Group_Eighties  Age Group_Fifties  \
Class
19071700                -0.968569           -0.077183           0.221351
19071700,40166274       -0.889907           -0.589265          -0.335165
```

40164929	0.194600	0.519966	-0.138669
40164930	0.420817	0.618676	0.309640
40166274	0.784469	-0.888983	-0.148979
Other	0.458589	0.416789	0.091822

	Age Group_Forties	Age Group_Seventies	Age Group_Sixties	\
Class				
19071700	0.382320	-0.202945	-0.123257	
19071700,40166274	0.493479	-0.430942	-0.395312	
40164929	0.144930	0.258276	0.374548	
40164930	-0.717196	0.096053	0.302286	
40166274	-0.522340	-0.331835	-0.342043	
Other	0.218806	0.611394	0.183779	

	Age Group_Teens	Age Group_Thirties	Age Group_Twenties	\
Class				
19071700	-0.172962	0.186572	0.756503	
19071700,40166274	0.788309	0.505485	0.856006	
40164929	-0.162396	-0.248012	-0.941639	
40164930	-0.057959	-0.102231	-0.873761	
40166274	-0.302294	0.687621	1.064813	
Other	-0.092697	-1.029434	-0.861922	

	Race_Asian	Race_Black	Race_Other	Race_White	\
Class					
19071700	0.093863	-0.012452	-0.125744	0.192047	
19071700,40166274	-0.851759	0.320316	0.367685	0.092576	
40164929	-0.124757	-0.133511	0.193953	0.521729	
40164930	-0.118452	-0.648549	1.136844	-0.589117	
40166274	1.138773	0.651920	-1.461457	-0.552708	
Other	-0.137669	-0.177725	-0.111281	0.335472	

	Race_White/ Hispanic	Gender_Female	Gender_Male	Intercept
Class				
19071700	-0.145884	-0.024036	0.025867	0.198079
19071700,40166274	0.073870	-0.051293	0.053982	0.227618
40164929	-0.455810	0.017013	-0.015410	0.203946
40164930	0.215597	-0.038802	0.035126	-0.368344
40166274	0.223900	0.176781	-0.176352	0.029854
Other	0.088327	-0.079663	0.076788	-0.291154