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 = 7
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
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]: 8

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

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

```
Accuracy: 0.09
Actual vs Predicted:
                 Actual          Predicted
408            19071700              Other
387            19071700           40164929
803            19071700           19077638
81                Other           40164929
942            19071700  19071700,40164929
…                     …                  …
596   19071700,40166274           40166274
1710              Other              Other
894   19071700,40164929           19077638
1226              Other  19071700,40164929
1466              Other           40166274

[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
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	-0.923121	-0.297700	0.291644
19071700,40164929	-0.693981	0.328933	-0.918028
19071700,40166274	-0.879620	-0.782580	-0.239998
19077638	0.846458	1.062979	0.453565
40164929	0.096192	0.288539	-0.061719
40164930	0.435910	0.352024	0.355013
40166274	0.682233	-1.080034	-0.064814
Other	0.435929	0.127837	0.184338

Class	Age Group_Forties	Age Group_Seventies	Age Group_Sixties \
19071700	0.332617	-0.368535	-0.219304
19071700,40164929	0.556370	0.833473	1.035745
19071700,40166274	0.465447	-0.567364	-0.466278
19077638	-0.399485	0.119819	-0.476138
40164929	0.120876	0.102137	0.290142
40164930	-0.721667	-0.061790	0.198423
40166274	-0.536977	-0.514216	-0.430184
Other	0.182819	0.456477	0.067594

Class	Age Group_Teens	Age Group_Thirties	Age Group_Twenties \
19071700	-0.135106	0.446760	0.872011
19071700,40164929	-0.026349	-0.698460	-0.418011
19071700,40166274	0.663650	0.800669	1.006034
19077638	-0.032600	-0.960568	-0.613838
40164929	-0.123574	0.011407	-0.724769
40164930	-0.043835	0.154622	-0.667522
40166274	-0.229986	0.967006	1.207459
Other	-0.072199	-0.721436	-0.661365

Class	Race_Asian	Race_Black	Race_Other	Race_White \
19071700	0.405875	0.106935	-0.098454	-0.184802
19071700,40164929	-0.756849	0.271881	-0.953314	0.906109
19071700,40166274	-0.550874	0.416034	0.400314	-0.241887
19077638	-1.108334	-0.767492	0.670938	1.137412
40164929	0.225987	-0.016686	0.257748	0.050522
40164930	0.160124	-0.584842	1.083477	-0.752746
40166274	1.344776	0.605128	-1.313211	-0.684583
Other	0.279294	-0.030958	-0.047497	-0.230024

Class	Race_White/ Hispanic	Gender_Female	Gender_Male	Intercept
19071700	-0.230289	0.034532	-0.035267	0.403213
19071700,40164929	0.531867	-0.310261	0.309954	-1.168825
19071700,40166274	-0.023625	0.011889	-0.011929	0.422937

19077638	0.067668	-0.072397	0.072590	-0.142634
40164929	-0.518340	0.080162	-0.080931	0.379521
40164930	0.095165	0.025614	-0.024435	-0.117992
40166274	0.048376	0.240431	-0.239945	0.333377
Other	0.029178	-0.009970	0.009964	-0.109598