MIMIC-III SELF-TUTORIAL

Patient Service Utilization Prediction

AI 395T | **AI in Healthcare** | Dr. Ying Ding

Presenter: **Cameron Joyce** (cbj746)

This slide deck covers:

Slides 0-1: Tutorial Introduction

1. Data Preprocessing

- Merge CSVs
- Handling Missing Values
- Encoding Categorical Features

2. Model Training

- Splitting Data
- Decision Tree Classifiers
- Training the Model

3. Model Evaluation

- More Accuracy Metrics
- Classification Report Analysis
- Interpretation of Model Results

4. Making Predictions

- Input Data Preparation
- Using the Trained Model
- Example Predictions

Introduction

What is patient service utilization and why does it matter?

Patient service utilization refers to how patients use healthcare services, including visits, treatments, and services. It matters because it helps healthcare providers allocate resources effectively, improve care quality by identifying trends and disparities, and manage costs by optimizing care delivery.

What data did I use?

MIMIC-III data. Specifically, I used patient and service information from the tables, PATIENTS, ADMISSIONS, AND SERVICES.

What are you predicting?

CURR_SERVICES, which are the services provided to the patient after admission. Understanding which services patients are likely to receive based on their diagnosis helps healthcare providers allocate resources effectively.

What model did I pick and why?

I chose to use a Decision Tree Classifier **(DTC)** model in this code because of its simplicity, interpretability, and ability to handle categorical data effectively. Decision trees are easy to understand, making it easier for me to explain the model and its predictions in the tutorial. Additionally, decision trees can handle categorical features like gender, language, and diagnosis, which are prevalent in healthcare data, making them a suitable choice for this project.

What libraries did you use?

I used Pandas for data processing, scikit-learn for ML, and tabulate for nicer visualizations

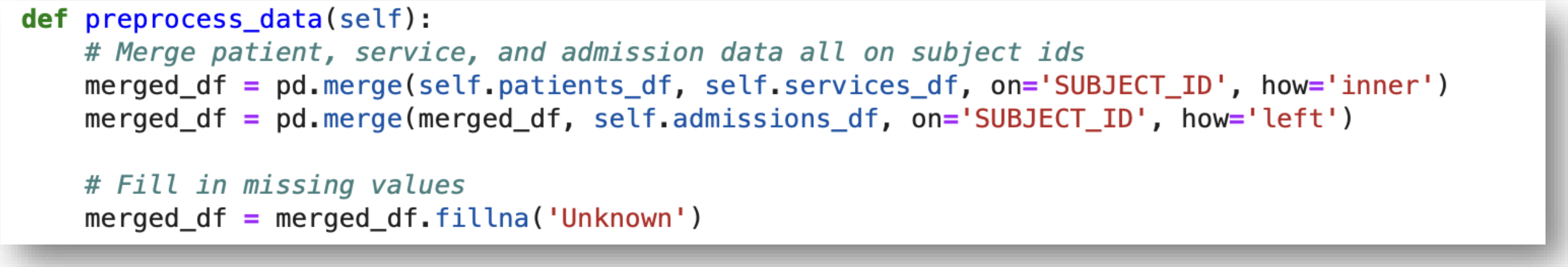
Where can I access this code?

https://github.com/CameronBJoyce/UTA_AI_HEALTHCARE/

Data Preprocessing

Merge CSVs and Handle Missing Values

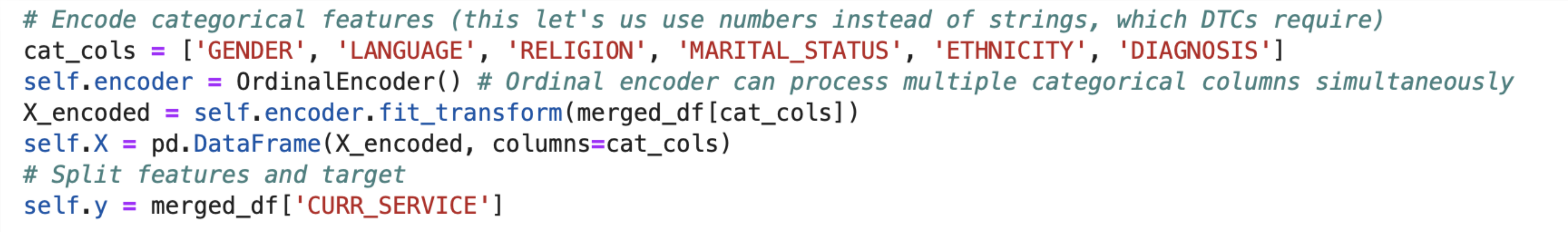
We needed this code to combine and clean patient, service, and admission data for analysis.



```
def preprocess_data(self):
    # Merge patient, service, and admission data all on subject ids
    merged_df = pd.merge(self.patients_df, self.services_df, on='SUBJECT_ID', how='inner')
    merged_df = pd.merge(merged_df, self.admissions_df, on='SUBJECT_ID', how='left')

    # Fill in missing values
    merged_df = merged_df.fillna('Unknown')
```

Encoding Categorical Features

This code converts categorical data into numerical format required by DTCs for model training.

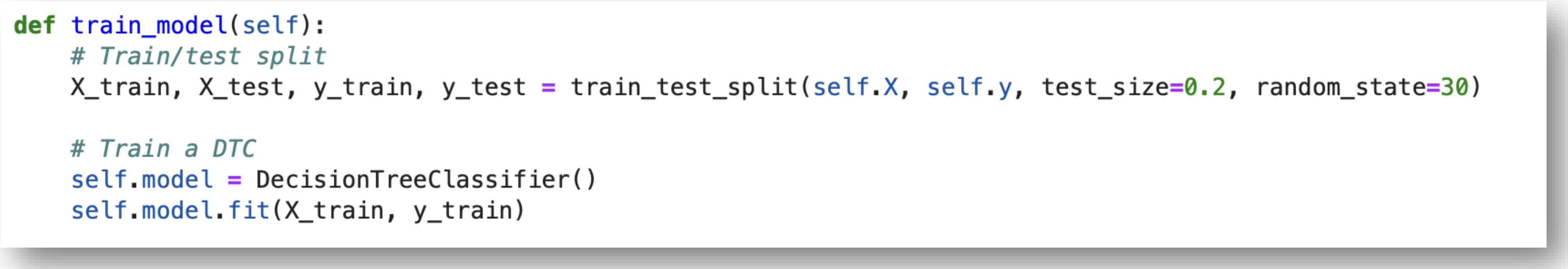


```
# Encode categorical features (this let's us use numbers instead of strings, which DTCs require)
cat_cols = ['GENDER', 'LANGUAGE', 'RELIGION', 'MARITAL_STATUS', 'ETHNICITY', 'DIAGNOSIS']
self.encoder = OrdinalEncoder() # Ordinal encoder can process multiple categorical columns simultaneously
X_encoded = self.encoder.fit_transform(merged_df[cat_cols])
self.X = pd.DataFrame(X_encoded, columns=cat_cols)
# Split features and target
self.y = merged_df['CURR_SERVICE']
```

Model Training

Splitting Data and DTC Training

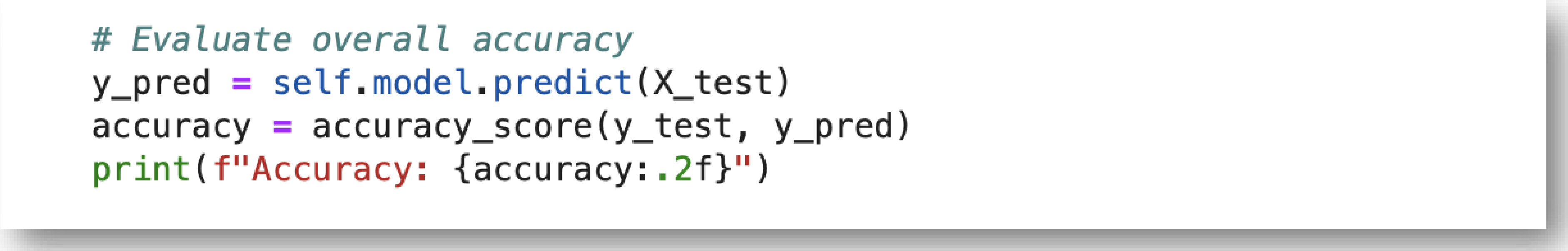
After encoding, we implement an 80/20 Train-Test split and train the model



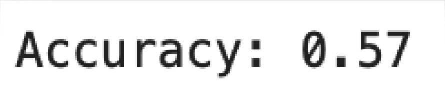
```python
def train_model(self):
    # Train/test split
    X_train, X_test, y_train, y_test = train_test_split(self.X, self.y, test_size=0.2, random_state=30)

    # Train a DTC
    self.model = DecisionTreeClassifier()
    self.model.fit(X_train, y_train)
```

Initial Accuracy Evaluation

This calculates and printing the accuracy of the model's predictions on the test data.



```python
    # Evaluate overall accuracy
    y_pred = self.model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    print(f"Accuracy: {accuracy:.2f}")
```



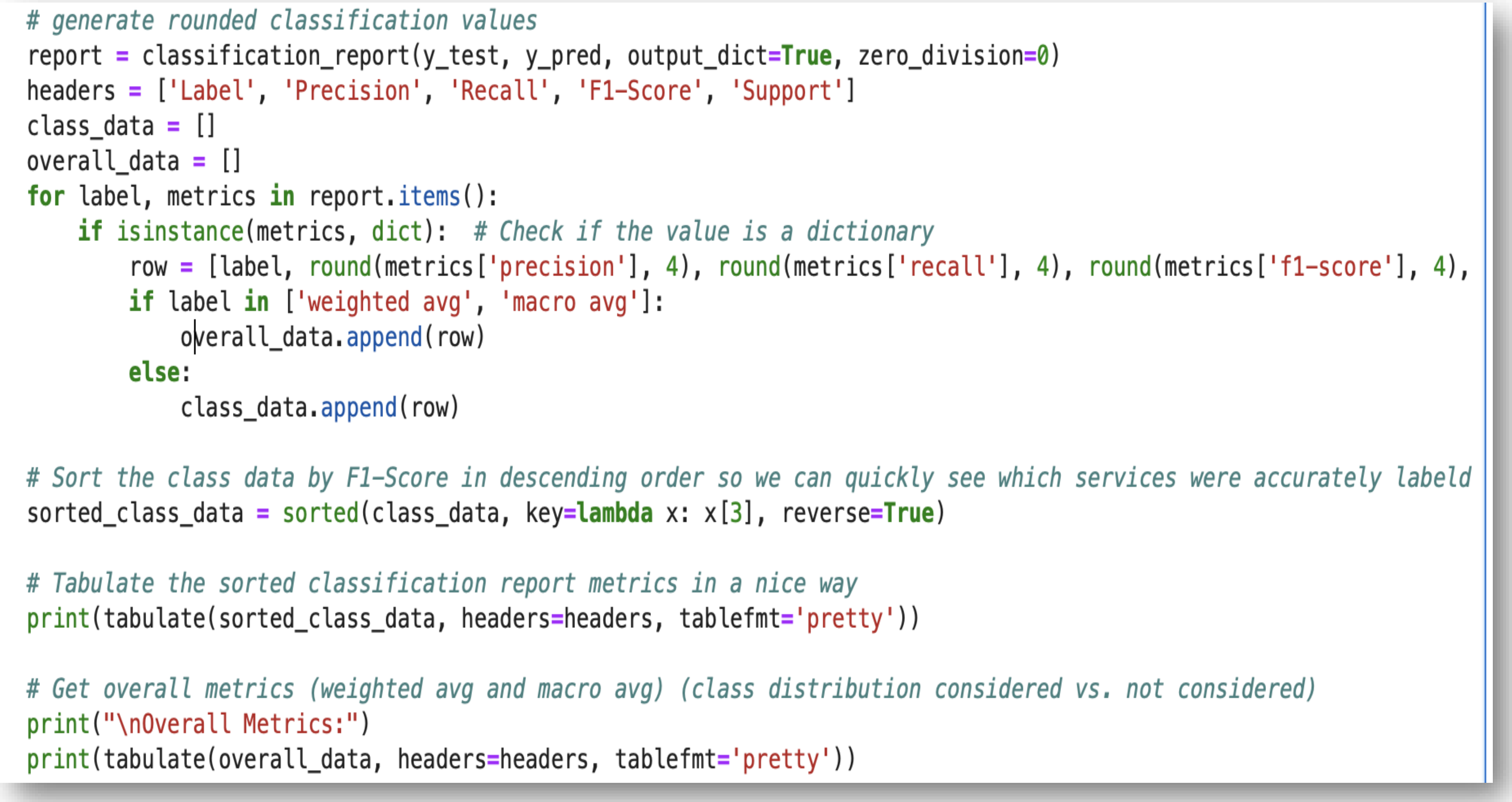
```
Accuracy: 0.57
```

Not great (I wouldn't want this to be my doctor), but not awful given the variety of services. More information in speakers notes.

Model Evaluation (Code)

More Metrics

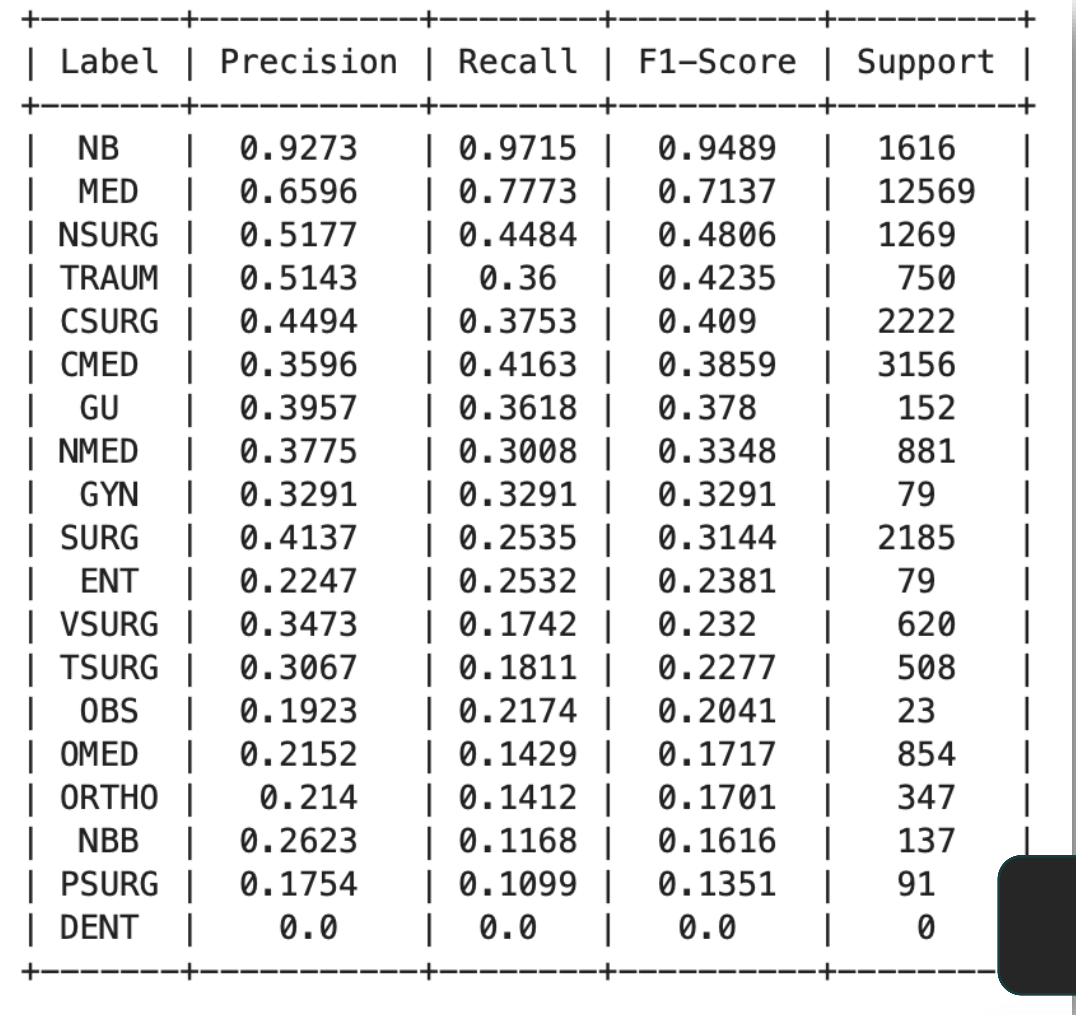
This code prints formatted classification report with precision, recall, F1-Score, and support metrics for each label, sorted by F1-Score, and displays overall metrics (weighted and macro).



```python
# generate rounded classification values
report = classification_report(y_test, y_pred, output_dict=True, zero_division=0)
headers = ['Label', 'Precision', 'Recall', 'F1-Score', 'Support']
class_data = []
overall_data = []
for label, metrics in report.items():
    if isinstance(metrics, dict):  # Check if the value is a dictionary
        row = [label, round(metrics['precision'], 4), round(metrics['recall'], 4), round(metrics['f1-score'], 4),
        if label in ['weighted avg', 'macro avg']:
            overall_data.append(row)
        else:
            class_data.append(row)

# Sort the class data by F1-Score in descending order so we can quickly see which services were accurately labeld
sorted_class_data = sorted(class_data, key=lambda x: x[3], reverse=True)

# Tabulate the sorted classification report metrics in a nice way
print(tabulate(sorted_class_data, headers=headers, tablefmt='pretty'))

# Get overall metrics (weighted avg and macro avg) (class distribution considered vs. not considered)
print("\nOverall Metrics:")
print(tabulate(overall_data, headers=headers, tablefmt='pretty'))
```

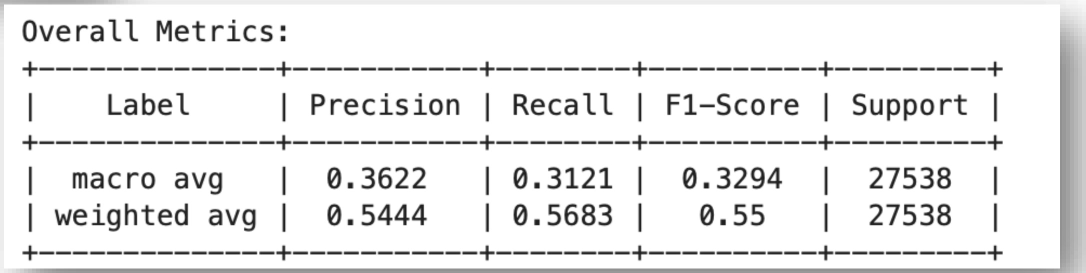
Model Evaluation (Results)

Classification Report Results



```
+-------+-----------+--------+----------+---------+
| Label | Precision | Recall | F1-Score | Support |
+-------+-----------+--------+----------+---------+
|  NB   |  0.9273   | 0.9715 |  0.9489  |  1616   |
|  MED  |  0.6596   | 0.7773 |  0.7137  |  12569  |
| NSURG |  0.5177   | 0.4484 |  0.4806  |  1269   |
| TRAUM |  0.5143   |  0.36  |  0.4235  |   750   |
| CSURG |  0.4494   | 0.3753 |  0.409   |  2222   |
| CMED  |  0.3596   | 0.4163 |  0.3859  |  3156   |
|  GU   |  0.3957   | 0.3618 |  0.378   |   152   |
| NMED  |  0.3775   | 0.3008 |  0.3348  |   881   |
|  GYN  |  0.3291   | 0.3291 |  0.3291  |   79    |
| SURG  |  0.4137   | 0.2535 |  0.3144  |  2185   |
|  ENT  |  0.2247   | 0.2532 |  0.2381  |   79    |
| VSURG |  0.3473   | 0.1742 |  0.232   |   620   |
| TSURG |  0.3067   | 0.1811 |  0.2277  |   508   |
|  OBS  |  0.1923   | 0.2174 |  0.2041  |   23    |
| OMED  |  0.2152   | 0.1429 |  0.1717  |   854   |
| ORTHO |   0.214   | 0.1412 |  0.1701  |   347   |
|  NBB  |  0.2623   | 0.1168 |  0.1616  |   137   |
| PSURG |  0.1754   | 0.1099 |  0.1351  |   91
| DENT  |    0.0    |  0.0   |   0.0    |    0
+-------+-----------+--------+----------+--------
```

Classification Report Results

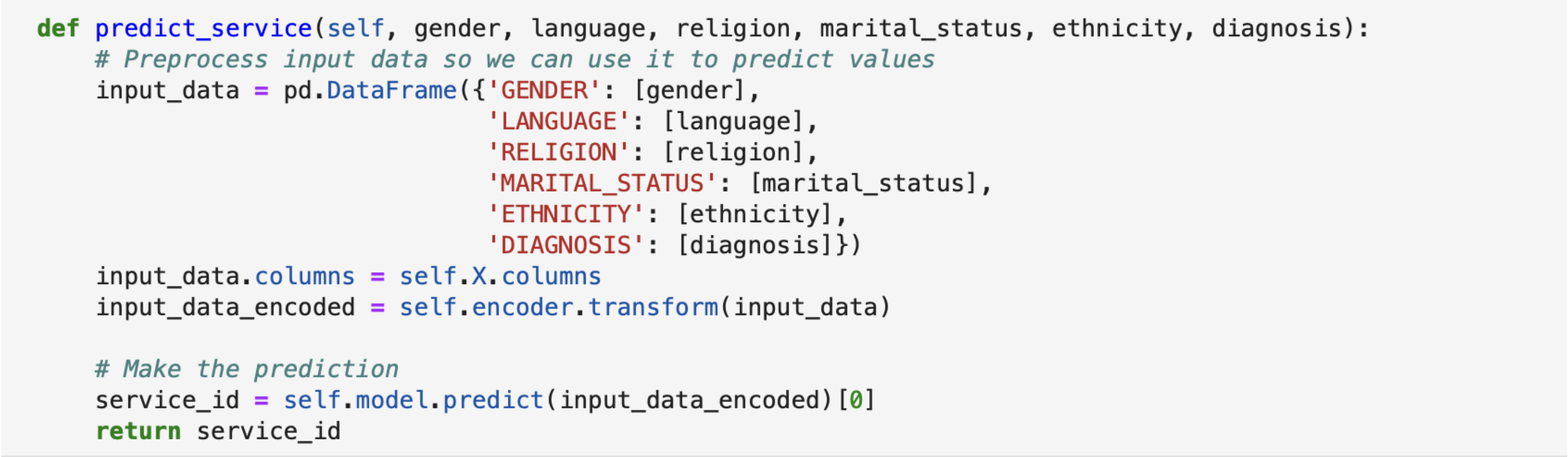


```
Overall Metrics:
+--------------+-----------+--------+----------+---------+
|    Label     | Precision | Recall | F1-Score | Support |
+--------------+-----------+--------+----------+---------+
|  macro avg   |  0.3622   | 0.3121 |  0.3294  |  27538  |
| weighted avg |  0.5444   | 0.5683 |   0.55   |  27538  |
+--------------+-----------+--------+----------+---------+
```

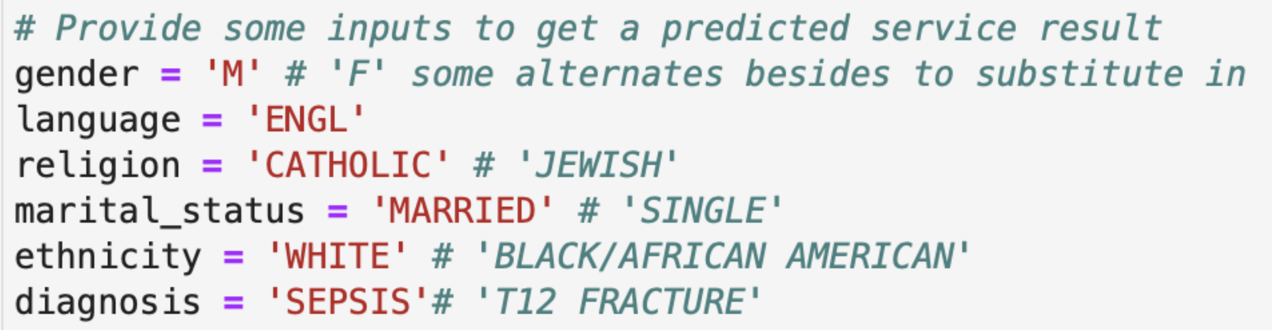
Making Predictions

Using the Trained Model

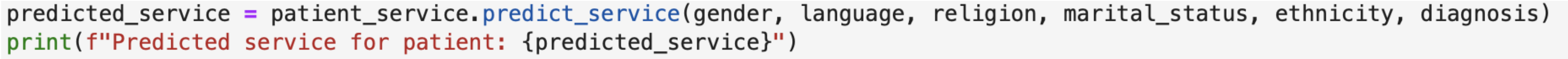
This code preprocesses input data and predicts a CURR_SERVICE using the trained DTC model.



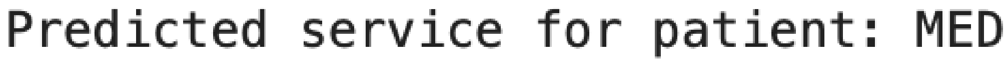
```
def predict_service(self, gender, language, religion, marital_status, ethnicity, diagnosis):
    # Preprocess input data so we can use it to predict values
    input_data = pd.DataFrame({'GENDER': [gender],
                               'LANGUAGE': [language],
                               'RELIGION': [religion],
                               'MARITAL_STATUS': [marital_status],
                               'ETHNICITY': [ethnicity],
                               'DIAGNOSIS': [diagnosis]})
    input_data.columns = self.X.columns
    input_data_encoded = self.encoder.transform(input_data)

    # Make the prediction
    service_id = self.model.predict(input_data_encoded)[0]
    return service_id
```



```
# Provide some inputs to get a predicted service result
gender = 'M' # 'F' some alternates besides to substitute in
language = 'ENGL'
religion = 'CATHOLIC' # 'JEWISH'
marital_status = 'MARRIED' # 'SINGLE'
ethnicity = 'WHITE' # 'BLACK/AFRICAN AMERICAN'
diagnosis = 'SEPSIS'# 'T12 FRACTURE'
```



```
predicted_service = patient_service.predict_service(gender, language, religion, marital_status, ethnicity, diagnosis)
print(f"Predicted service for patient: {predicted_service}")
```



```
Predicted service for patient: MED
```

Thank you!

Please check out the code at :https://github.com/CameronBJoyce/UTA_AI_HEALTHCARE/