UDACITY

‹ Return to Classroom

AI and EHR

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

Udacity student,

your project shows how committed you are to the course. Working with AI and Healthcare data is challenging and also important for future researches and companies to enhance this area.

Keep the great work on the next sections of this amazing Udacity course!!! I would like to share some nice resources:

A guide to start your path in Data Science and Machine Learning:

How can Artificial Intelligence Help Health Care?

AI and machine learning for healthcare

Finally, I would like to share interview questions tips:

Data Science Interview

😄

Exploratory Data Analysis

The project correctly identified which field(s) has/have a high amount of missing/zero values.

The project correctly identified which field(s) has/have a Gaussian distribution shape based on the histogram analysis.

The project correctly identified fields with high cardinality.

The project justified why these fields had high cardinality.

The project correctly describes the distributions for age and gender.

Optional: The project uses Tensorflow Data Validation Visualizations to analyze which fields have a high amount of missing/null values or high cardinality.

Exploratory Data Analysis

Great job! 😄

In this part of the project, you are asked to:

[✓] - The project correctly identified which field(s) has/have a high amount of missing/zero values.
[✓] - The project correctly identified which field(s) has/have a Gaussian distribution shape based on the histogram analysis.
[✓] - The project correctly identified fields with high cardinality.
[✓] - The project justified why these fields had high cardinality.
[✓] - The project correctly describes the distributions for age and gender.

The project correctly identifies whether to include/exclude payer_code and weight fields.

The project justified why these fields should be included/excluded by using supporting data analysis.

Exploratory Data Analysis

Awesome! 😄

The project correctly identifies whether to include/exclude payer_code and weight fields. You also justified why these fields should be included/excluded by using supporting data analysis.

Data Preparation

The project uses the correct level(s) for the given EHR dataset (line, encounter, patient) and transforms, aggregates and filters appropriately.

Data Preparation

Good job! 😄

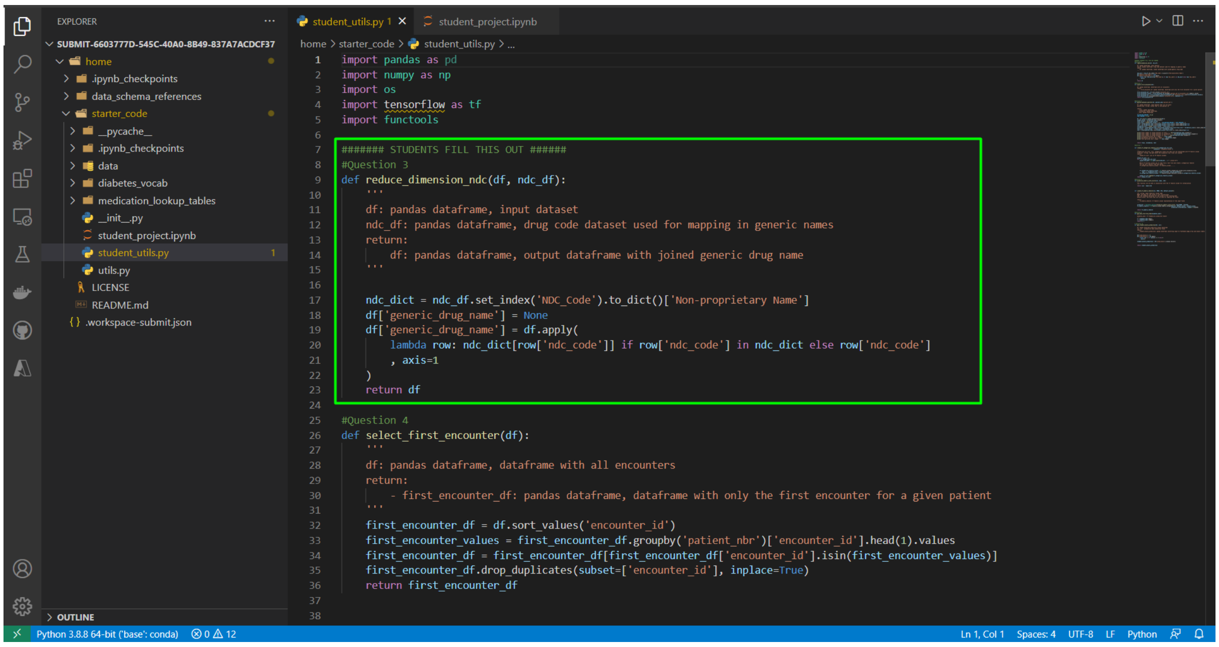
You project uses the correct level(s) for the EHR dataset. One way to validate is the **passed test** in your notebook! 😉

The project correctly maps NDC codes to generic drug names and prints out the correct mappings in the notebook.

Data Preparation

Perfect! 😄

Your file `student_utils.py` is completed and have all necessary implementations. The function is working correctly and you have printed the output.



```python
import pandas as pd
import numpy as np
import os
import tensorflow as tf
import functools

####### STUDENTS FILL THIS OUT ######
#Question 3
def reduce_dimension_ndc(df, ndc_df):
    '''
    df: pandas dataframe, input dataset
    ndc_df: pandas dataframe, drug code dataset used for mapping in generic names
    return:
        df: pandas dataframe, output dataframe with joined generic drug name
    '''


    ndc_dict = ndc_df.set_index('NDC_Code').to_dict()['Non-proprietary Name']
    df['generic_drug_name'] = None
    df['generic_drug_name'] = df.apply(
        lambda row: ndc_dict[row['ndc_code']] if row['ndc_code'] in ndc_dict else row['ndc_code']
        , axis=1
    )
    return df

#Question 4
def select_first_encounter(df):
    '''
    df: pandas dataframe, dataframe with all encounters
    return:
        - first_encounter_df: pandas dataframe, dataframe with only the first encounter for a given patient
    '''
    first_encounter_df = df.sort_values('encounter_id')
    first_encounter_values = first_encounter_df.groupby('patient_nbr')['encounter_id'].head(1).values
    first_encounter_df = first_encounter_df[first_encounter_df['encounter_id'].isin(first_encounter_values)]
    first_encounter_df.drop_duplicates(subset=['encounter_id'], inplace=True)
    return first_encounter_df


```

The project has correctly split the original dataset into train, validation, and test datasets.

The projects dataset splits do not contain patient or encounter data leakage.

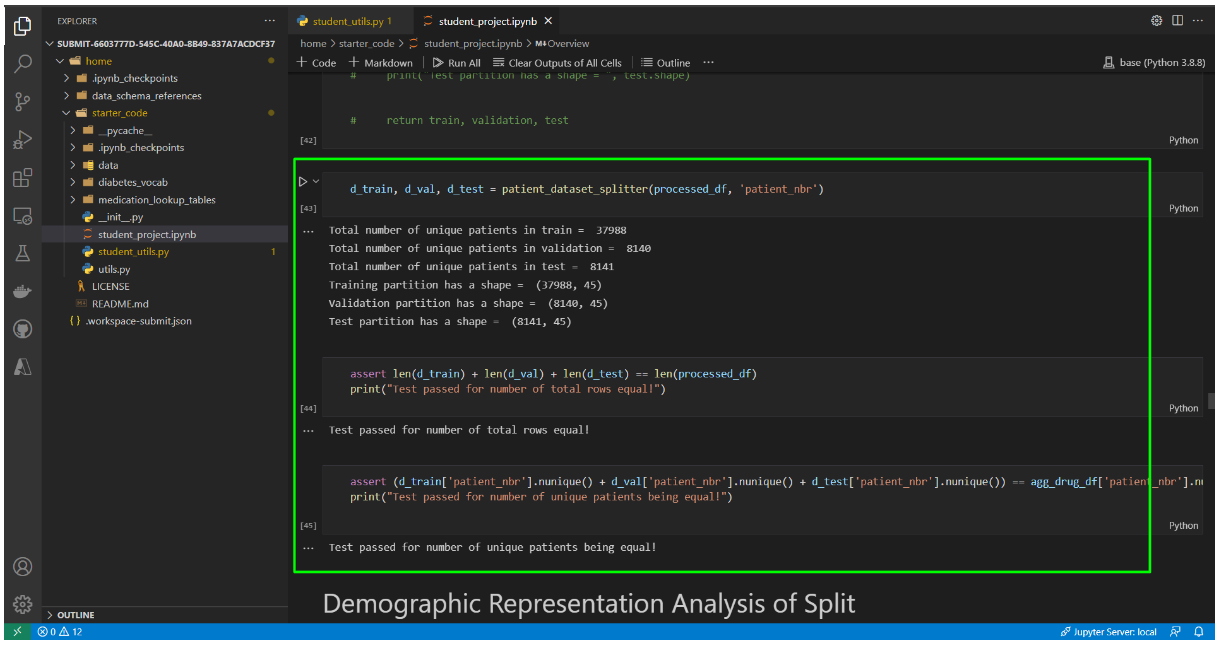
The Projects code passes the Encounter Test.

Data Preparation

All set here! 😄

This part of the project you have successfully completed! The dataset has a train, validation and test sets.

Notebook



```
#     print("Test partition has a shape = ", test.shape)

#     return train, validation, test
```

```
d_train, d_val, d_test = patient_dataset_splitter(processed_df, 'patient_nbr')
```

```
Total number of unique patients in train =  37988
Total number of unique patients in validation =  8140
Total number of unique patients in test =  8141
Training partition has a shape =  (37988, 45)
Validation partition has a shape =  (8140, 45)
Test partition has a shape =  (8141, 45)
```

```
assert len(d_train) + len(d_val) + len(d_test) == len(processed_df)
print("Test passed for number of total rows equal!")
```

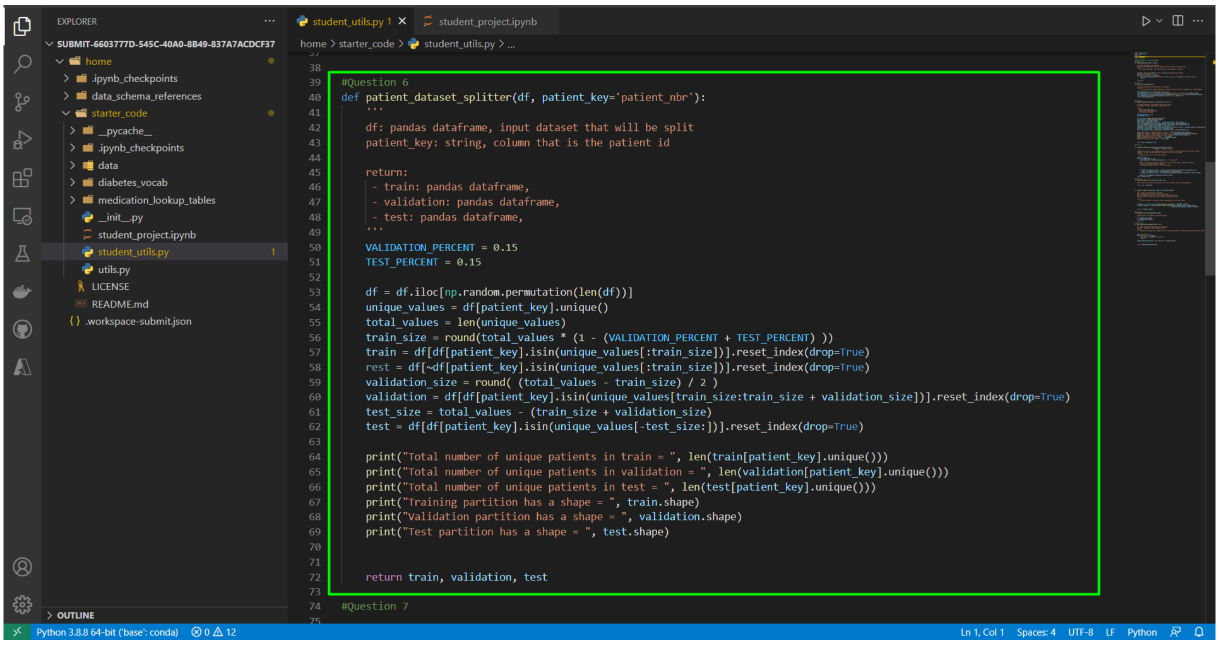
```
Test passed for number of total rows equal!
```

```
assert (d_train['patient_nbr'].nunique() + d_val['patient_nbr'].nunique() + d_test['patient_nbr'].nunique()) == agg_drug_df['patient_nbr'].nu
print("Test passed for number of unique patients being equal!")
```

```
Test passed for number of unique patients being equal!
```

Demographic Representation Analysis of Split

Function



```
#Question 6
def patient_dataset_splitter(df, patient_key='patient_nbr'):
    '''
    df: pandas dataframe, input dataset that will be split
    patient_key: string, column that is the patient id

    return:
     - train: pandas dataframe,
     - validation: pandas dataframe,
     - test: pandas dataframe,
    '''
    VALIDATION_PERCENT = 0.15
    TEST_PERCENT = 0.15

    df = df.iloc[np.random.permutation(len(df))]
    unique_values = df[patient_key].unique()
    total_values = len(unique_values)
    train_size = round(total_values * (1 - (VALIDATION_PERCENT + TEST_PERCENT) ))
    train = df[df[patient_key].isin(unique_values[:train_size])].reset_index(drop=True)
    rest = df[~df[patient_key].isin(unique_values[:train_size])].reset_index(drop=True)
    validation_size = round( (total_values - train_size) / 2 )
    validation = df[df[patient_key].isin(unique_values[train_size:train_size + validation_size])].reset_index(drop=True)
    test_size = total_values - (train_size + validation_size)
    test = df[df[patient_key].isin(unique_values[-test_size:])].reset_index(drop=True)

    print("Total number of unique patients in train = ", len(train[patient_key].unique()))
    print("Total number of unique patients in validation = ", len(validation[patient_key].unique()))
    print("Total number of unique patients in test = ", len(test[patient_key].unique()))
    print("Training partition has a shape = ", train.shape)
    print("Validation partition has a shape = ", validation.shape)
    print("Test partition has a shape = ", test.shape)


    return train, validation, test

#Question 7
```

Feature Engineering

The project correctly completes the categorical feature transformer boilerplate function.

The project successfully uses this function to transform the demo dataset with at least one new categorical feature.

Feature Engineering

Great! 😉

The project successfully uses this function to transform the demo dataset with at least one new categorical feature.

The project correctly completes the numerical feature transformer boilerplate function.

The project successfully uses this function to transform the demo dataset with at least one new numerical feature.

The project's transformer function correctly incorporates the provided z-score normalizer function for normalization or another custom normalizer.

Feature Engineering

Good job here as well! 😉

The project successfully uses functions to transform the demo dataset with at least one new numerical feature.

Model Building and Analysis

The project has prepared the regression model predictions for TF Probability and binary classification outputs by doing the following:
Correctly utilized TF Probability to provide mean and standard deviation prediction outputs
Created an output prediction dataset that has the labels correctly mapped to a binary prediction and actual value.

Model Building and Analysis

Great job! 😄

Once again your project set the regression model for TF Probability and binary classification outputs. You also created the dataset with the necessary outputs.

The model has been evaluated across the following classification metrics: AUC, F1, precision, and recall.

Students have completed both questions for the model summary and address bias-variance tradeoff in regard to this problem.

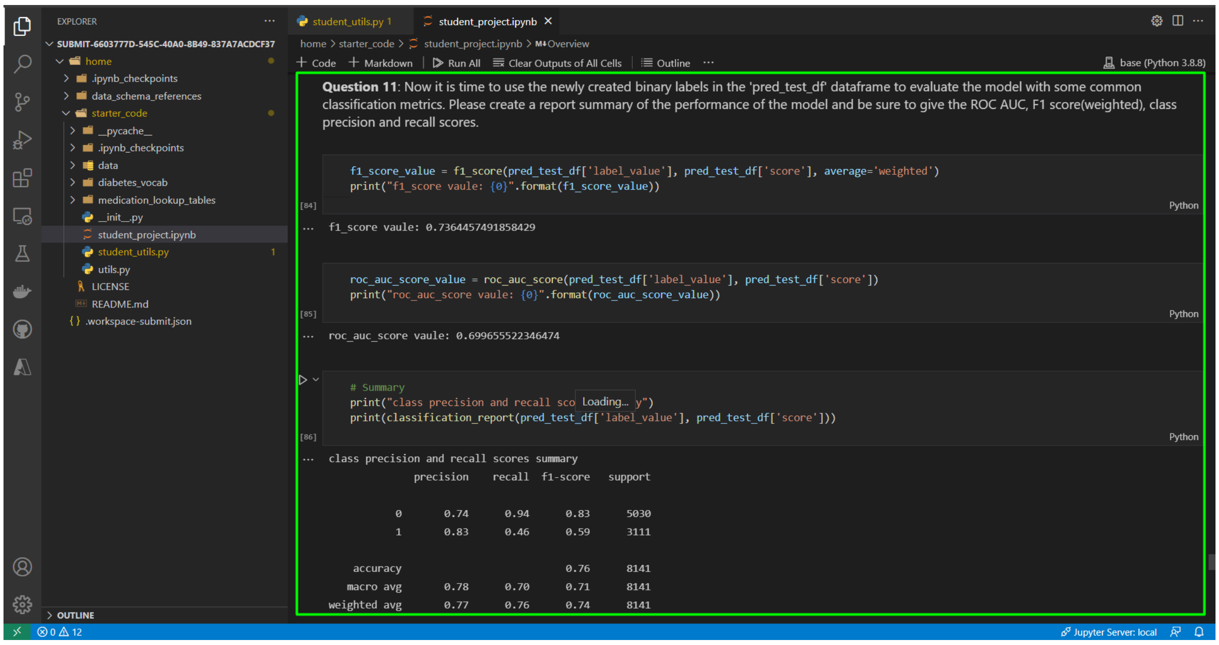
Model Building and Analysis

Nice job! 😄

Your model has been evaluated and you have presented the necessary metrics like AUC, F1, precision and recall.

As a suggestion, you can also use the ROC curve to improve your project

Notebook



Question 11: Now it is time to use the newly created binary labels in the 'pred_test_df' dataframe to evaluate the model with some common classification metrics. Please create a report summary of the performance of the model and be sure to give the ROC AUC, F1 score(weighted), class precision and recall scores.

```
[84]
f1_score_value = f1_score(pred_test_df['label_value'], pred_test_df['score'], average='weighted')
print("f1_score vaule: {0}".format(f1_score_value))
```

```
f1_score vaule: 0.7364457491858429
```

```
[85]
roc_auc_score_value = roc_auc_score(pred_test_df['label_value'], pred_test_df['score'])
print("roc_auc_score vaule: {0}".format(roc_auc_score_value))
```

```
roc_auc_score vaule: 0.699655522346474
```

```
[86]
# Summary
print("class precision and recall sco Loading... y")
print(classification_report(pred_test_df['label_value'], pred_test_df['score']))
```

```
class precision and recall scores summary
              precision    recall  f1-score   support

           0       0.74      0.94      0.83      5030
           1       0.83      0.46      0.59      3111

    accuracy                           0.76      8141
   macro avg       0.78      0.70      0.71      8141
weighted avg       0.77      0.76      0.74      8141
```

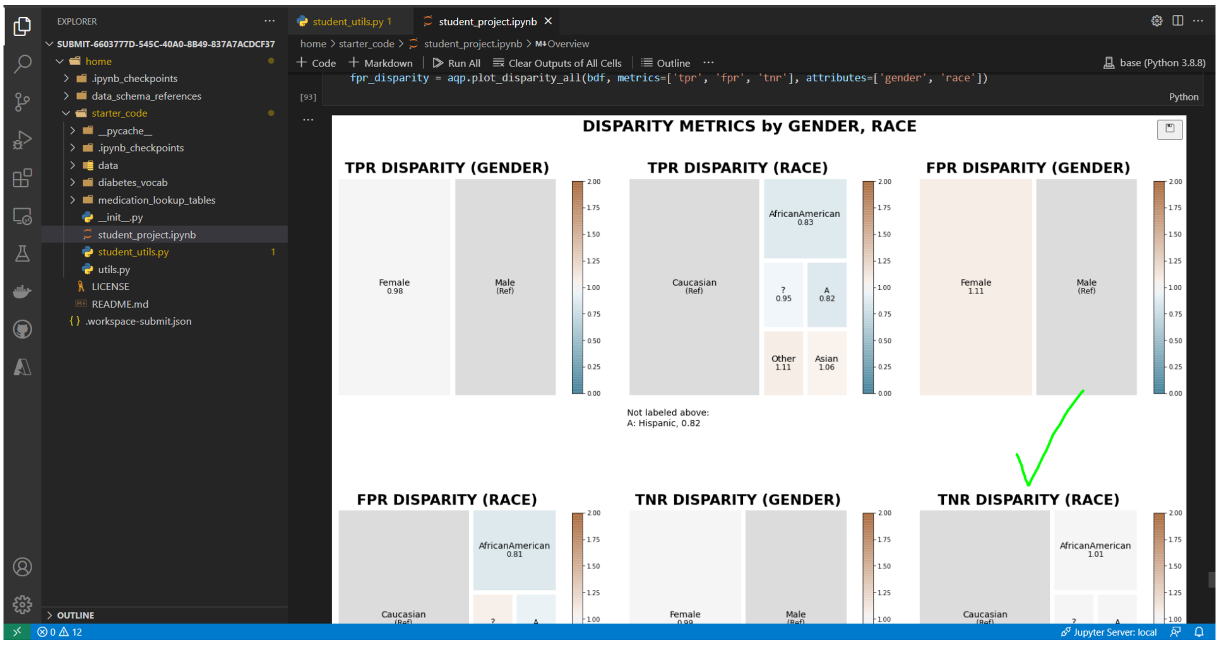
The project contains a bias report with the following:

- A visualization of at least two key metric(s) for patient selection
- A visualization showing at least one reference group fairness example and its comparison on at least one metric (e.g. TPR).
- Justification for analysis made about at least one visualization

Model Building and Analysis

Good job on this last part of your project! 😄

You have provided the key metrics for patient selection and justified your point of view!



```
fpr_disparity = aqp.plot_disparity_all(bdf, metrics=['tpr', 'fpr', 'tnr'], attributes=['gender', 'race'])
```

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review

START