Assignment 2
Introduction to Artificial Intelligence and Logic Programming
Prepared by: Dr. Ruba Alomari

Instructions

- This is an individual assignment.
- Do not share this assignment document or upload it to online sharing websites. Doing so violates the academic integrity policy.
- Copying another person's work is a breach of the academic integrity policy. Credit the source and author of any code you reuse.

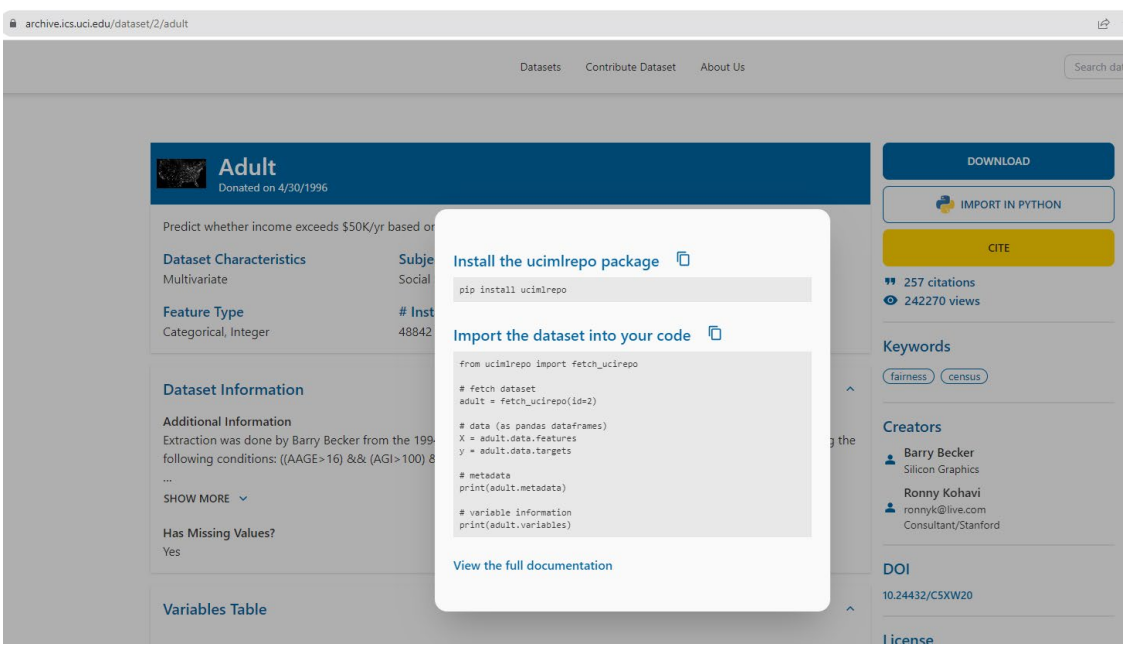
Dataset

Use the Adult training dataset available at https://archive.ics.uci.edu/dataset/2/adult to predict whether an adult's income exceeds $50K/yr based on census data.

Before proceeding with this assignment, read about the adult dataset and get yourself familiar with the dataset information from the above link.

Tasks Each of the below tasks is worth 10 marks, if a task has subtasks, the weight of the main task is divided on the subtasks as well.

Task 1 (10 Marks): Import the adult dataset from the `ucimlrepo`. Check the IMPORT IN PYTHON option provided by UCI for instructions on how to do so:



Note that at no point in this assignment, should you save the dataset locally to your machine. Doing so will result in zero marks for this assignment.

Use the code provided in the screenshot above to load the X and y from the adult.data.features and adult.data.targets respectively.

Task 2 (10 Marks): Take a quick look at the data structure (i.e., X) using `.head()`, `.info()`, `.describe()`, and `.shape`.

Tip: Shape should be (48842, 14))

Task 2.1: Plot a histogram of the data.

Task 3 (10 Marks): There are missing values in this dataset that are entered as **?**, check for the number of these missing values.

Tip: The number of missing values should be similar to the ones shown below.

```
age                 0
workclass        1836
fnlwgt              0
education           0
education-num       0
marital-status      0
occupation       1843
relationship        0
race                0
sex                 0
capital-gain        0
capital-loss        0
hours-per-week      0
native-country    583
dtype: int64
```

Task 4 (10 Marks): Replace the missing values you found in the previous step with null (nan). Run a `X.info()` to see the non-null count.

Tip: Your `X.info()` should look similar to the one below at this step:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48842 entries, 0 to 48841
Data columns (total 14 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             48842 non-null  int64
 1   workclass       46043 non-null  object
 2   fnlwgt          48842 non-null  int64
 3   education       48842 non-null  object
 4   education-num   48842 non-null  int64
 5   marital-status  48842 non-null  object
 6   occupation      46033 non-null  object
 7   relationship    48842 non-null  object
 8   race            48842 non-null  object
 9   sex             48842 non-null  object
 10  capital-gain    48842 non-null  int64
 11  capital-loss    48842 non-null  int64
 12  hours-per-week  48842 non-null  int64
 13  native-country  47985 non-null  object
dtypes: int64(6), object(8)
memory usage: 5.2+ MB
```

Task 5 (10 Marks): Create and apply a preprocessing pipeline to:

1. Fill in the missing numerical values with the mean using a `SimpleImputer`.
2. Scale the numerical columns using `StandardScaler`. Do not scale the target.
3. Fill in the missing categorical values with the `most_frequent` value using `SimpleImputer`.
4. Encode the categorical columns using `OneHotEncoder`. Do not encode the target.

- Display your pipeline.
- Print `X_prepared.shape`.

Tips:

- If you are facing an issue with the preprocessing pipeline producing a sparse matrix, pass a "`sparse_output=False`" option to the `OneHotEncoder` in the pipeline, i.e., `OneHotEncoder(sparse_output=False)`
- `X_prepared.shape` should be (48842, 105) at this point.

Task 6 (10 Marks): Check the target `value_counts`. You will notice that the target needs some data cleaning.

Task 7(10 Marks):: Remove the period at the end of the >50K. and <=50K. i.e., replace all instances that are <=50K. with <=50K , and replace all the instances that are >50K. with >50K

Run the `value_counts` again.

Tip: At this point, `value_counts` should be:

```
income
```

```
<=50K      37155
>50K       11687
```

Task 8 (10 Marks): Split the data into 80% training set and 20% testing set, print the shape of `X_train, X_test, y_train, y_test` in one command.

Tip: Shapes should be (39073, 105) (39073, 1) (9769, 105) (9769, 1)

Task 9 (10 Marks): Train a `svm` model (`svc`) to predict if the income of the adult exceeds 50K on the training set using: `kernel = poly, gamma = 1,` and `C =0.1`. Call your model `model_svm`.

Tip: If your model is taking a long time to train, train on the first 10,000 examples only:

```
model_svm = SVC(kernel='poly', C=0.1, gamma=1)
model_svm.fit(X_train.iloc[:10000], y_train.iloc[:10000].values.ravel())
```

Task 9.1: Test your model on the X_Test, and report the `classification_report` on the `y_test` and `y_predict`.

Task 9.2: Display the confusion matrix of your test results using `ConfusionMatrixDisplay.from_predictions(y_test, y_predict)`

Task 10 (10 Marks): Use `GridSearchCV` to find the best value of kernel, gamma, and C.

Task 10.1: Split the dataset into 60% training, 20% validation, and 20% testing. Use the code below to perform the split:

```
X_train, X_validation_test, y_train, y_validation_test = train_test_split(X_prepared, y, test_size=0.4, random_state=42)

X_validation, X_test, y_validation, y_test = train_test_split(X_validation_test, y_validation_test, test_size=0.5, random_state=42)

print(X_train.shape, y_train.shape, X_validation.shape, y_validation.shape, X_test.shape, y_test.shape)
```

[27] ✓ 0.0s

```
(29305, 105) (29305, 1) (9768, 105) (9768, 1) (9769, 105) (9769, 1)
```

Tip: Shapes should be `(29305, 105) (29305, 1) (9768, 105) (9768, 1) (9769, 105) (9769, 1)`

Task 10.2: Use the below code snippet to pass the following hyperparameters for the `GridSearchCV` to find the best ones:

```
# code author luisguiserrano

from sklearn.model_selection import GridSearchCV

svm_parameters = {'kernel': ['rbf'],
                  'C': [0.01, 0.1, 1 , 10],
                  'gamma': [0.01, 1, 10]
                 }
svm = SVC()
svm_gs = GridSearchCV(estimator = svm,
                      param_grid = svm_parameters)
svm_gs.fit(X_train.iloc[:10000], y_train.iloc[:10000].values.ravel())

svm_winner = svm_gs.best_estimator_
svm_winner.score(X_validation, y_validation)
```

4m 28.8s

Task 10.2: Check the svm winner parameters using `svm_winner`

Submission:

Submit a url to your notebook, as well as your notebook using the following instructions:

1- Run your notebook, and save it with the results.
2- Upload your notebook to your github. Submit the url to your github notebook in eclass. In the submission dropbox on eclass there is a space for you to paste your url.
3- Submit to eclass your source code notebook **.ipynb** with the code already run and the output results included and saved in your notebook. Note that the dataset in your notebook should be loaded from the sklearn repository. Do not save the dataset to or load it from your local drive.
4- Include assignment # and your name in your eclass submitted notebook in the following format: A2-FirstName-LastName.ipynb.
5- In your notebook make sure to list:
    a. The question/task you are answering in a markdown cell before each code section (similar to the end-to-end example posted to eclass)
    b. Comment your code.
    c. Submit a clean notebook free of errors and do not include any extra commands that are not needed to perform the tasks:
        i. Extra commands that are not necessary to perform the task will result in 5 marks deduction per extra command.
        ii. Errors in the output cells will result in 5 marks deduction per error.
        iii. Write efficient code.