UDACITY

‹ Return to Classroom

Disaster Response Pipeline

REVIEW

HISTORY

Requires Changes

2 specifications require changes

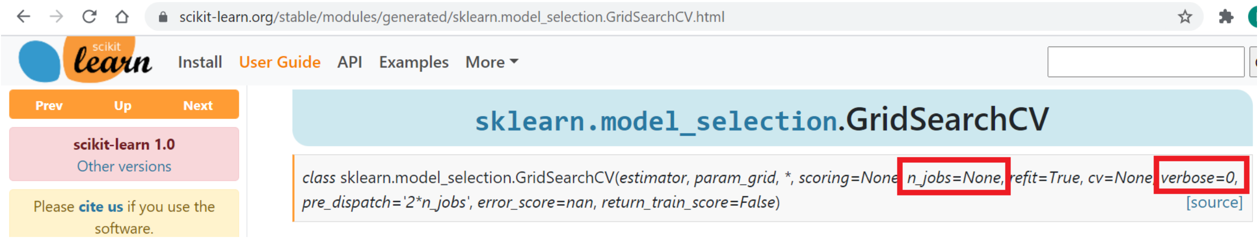
Dear Learner,

Awesome work is done in this submission. You have completed the tasks amazingly well.
Overall brilliant work.

You almost completed the project, few minor additions, which I am sure you can fix in no time.
Kindly follow the comments in the corresponding specifications.

I believe you will have further fun and a learning experience working through them.
Feel free to post your specific doubts at "https://knowledge.udacity.com/"

As far as your doubt is concerned, one of the ways to improve the time is to use the n_jobs parameter(n_jobs=-1) to use all the cores of the CPU. You can also make use of the verbose parameter(verbose=4) to visualize what is happening in the background.


scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
scikit learn
Install
User Guide
API
Examples
More
Prev
Up
Next
scikit-learn 1.0
Other versions
Please cite us if you use the software.

sklearn.model_selection.GridSearchCV

class sklearn.model_selection.GridSearchCV(*estimator*, *param_grid*, ***, *scoring=None*, *n_jobs=None*, *refit=True*, *cv=None*, *verbose=0*, *pre_dispatch='2*n_jobs'*, *error_score=nan*, *return_train_score=False*) [source]

You can refer to the document at- https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

Keep learning.
Good luck !!!

Github & Code Quality

✓

All project code is stored in a GitHub repository and a link to the repository has been provided for reviewers. The student made at least 3 commits to this repository.

Awesome, All project code is pushed to your GitHub repository.

✓

The README file includes a summary of the project, how to run the Python scripts and web app, and an explanation of the files in the repository. Comments are used effectively and each function has a docstring.

Perfect, your README file is properly documented mentioning about every step very clearly.

✓

Scripts have an intuitive, easy-to-follow structure with code separated into logical functions. Naming for variables and functions follows the PEP8 style guidelines.

Nice!,
Scripts have an intuitive, easy-to-follow structure with code separated into logical functions.
It's nice to see that you have used proper "docstrings" into your functions.

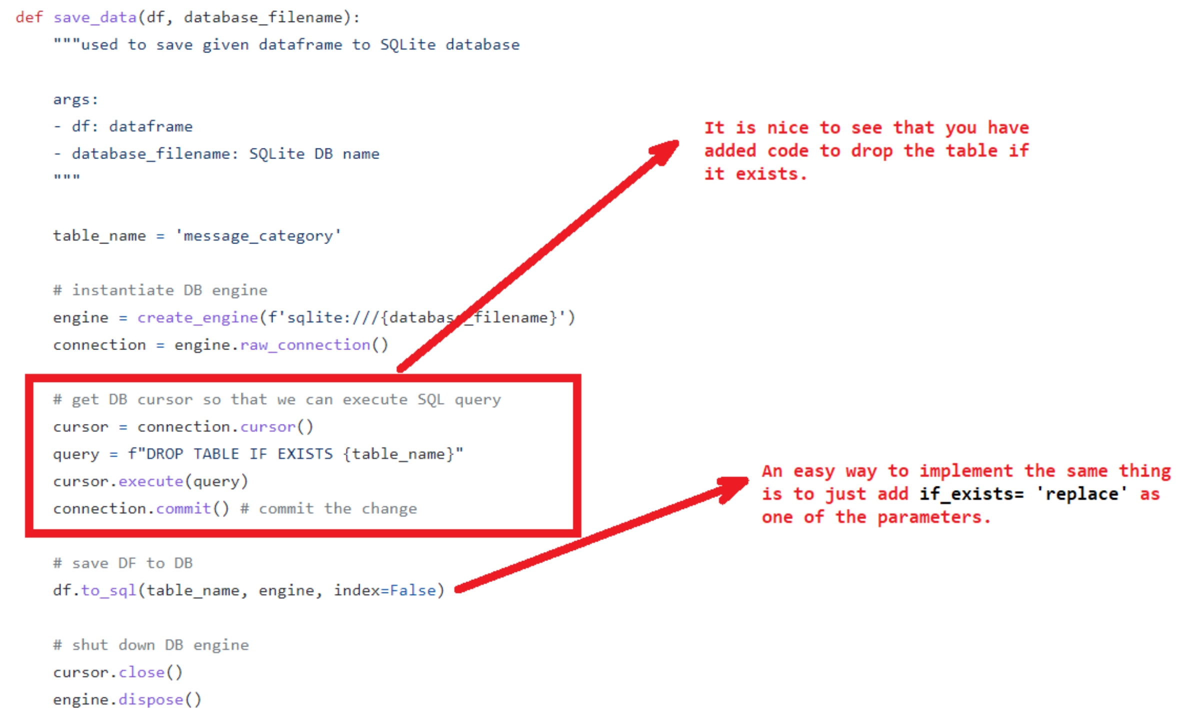
ETL

✓

The ETL script, process_data.py, runs in the terminal without errors. The script takes the file paths of the two datasets and database, cleans the datasets, and stores the clean data into a SQLite database in the specified database file path.

Yes, The ETL script, process_data.py, runs in the terminal without errors accepting required arguments.

Recommendation



```
def save_data(df, database_filename):
    """used to save given dataframe to SQLite database

    args:
    - df: dataframe
    - database_filename: SQLite DB name
    """

    table_name = 'message_category'

    # instantiate DB engine
    engine = create_engine(f'sqlite:///{database_filename}')
    connection = engine.raw_connection()

    # get DB cursor so that we can execute SQL query
    cursor = connection.cursor()
    query = f"DROP TABLE IF EXISTS {table_name}"
    cursor.execute(query)
    connection.commit() # commit the change

    # save DF to DB
    df.to_sql(table_name, engine, index=False)

    # shut down DB engine
    cursor.close()
    engine.dispose()
```

It is nice to see that you have added code to drop the table if it exists.

An easy way to implement the same thing is to just add **if_exists= 'replace'** as one of the parameters.

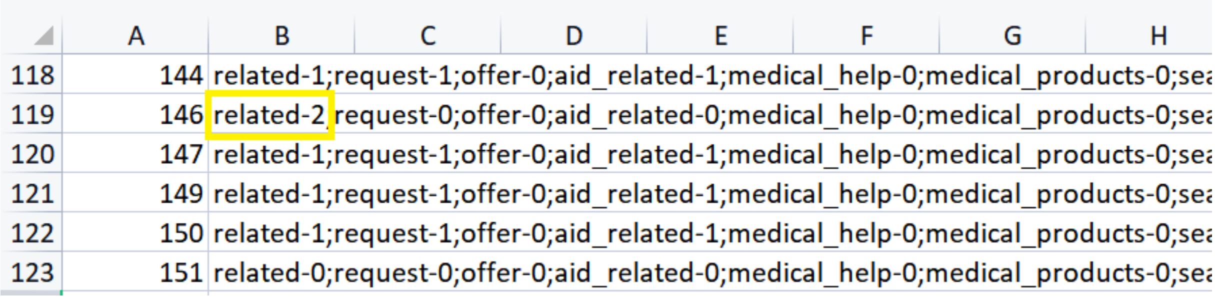
The script successfully follows steps to clean the dataset. It merges the messages and categories datasets, splits the categories column into separate, clearly named columns, converts values to binary, and drops duplicates.

- It merges the messages and categories datasets ✅
- Splits the categories column into separate, clearly named columns ✅
- Converts values to binary ❌
- Drops duplicates ✅

Explanation

The values in your cleaned data_frame should contain only binary values(i.e either 0 or 1).
But if you watch closely the "categories" column into the disaster_categories.csv file. You will find that certain "related" values are "related-2" which doesn't make any sense.

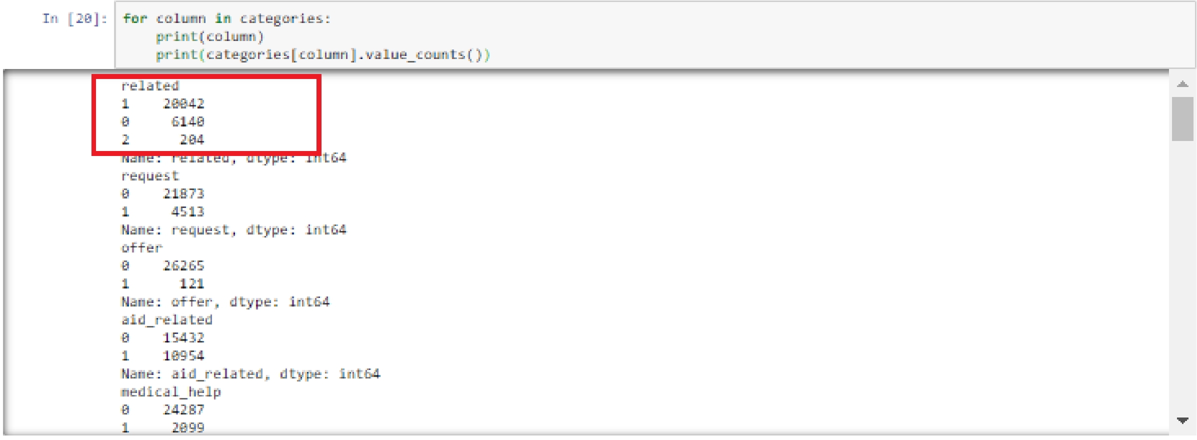
Snippet from disaster_categories.csv:-



	A	B	C	D	E	F	G	H
118	144	related-1;request-1;offer-0;aid_related-1;medical_help-0;medical_products-0;sea						
119	146	related-2;request-0;offer-0;aid_related-0;medical_help-0;medical_products-0;sea						
120	147	related-1;request-1;offer-0;aid_related-1;medical_help-0;medical_products-0;sea						
121	149	related-1;request-1;offer-0;aid_related-1;medical_help-0;medical_products-0;sea						
122	150	related-1;request-1;offer-0;aid_related-1;medical_help-0;medical_products-0;sea						
123	151	related-0;request-0;offer-0;aid_related-0;medical_help-0;medical_products-0;sea						

Hint

You can check for different columns something like:-



```
In [20]: for column in categories:
             print(column)
             print(categories[column].value_counts())

related
1    20042
0     6140
2      204
Name: related, dtype: int64
request
0    21873
1     4513
Name: request, dtype: int64
offer
0    26265
1      121
Name: offer, dtype: int64
aid_related
0    15432
1    10954
Name: aid_related, dtype: int64
medical_help
0    24287
1     2099
```

Kindly clean that. You can drop the entire row as the cleaning process.

Machine Learning

✓

The machine learning script, train_classifier.py, runs in the terminal without errors. The script takes the database file path and model file path, creates and trains a classifier, and stores the classifier into a pickle file to the specified model file path.

✓

The script uses a custom tokenize function using nltk to case normalize, lemmatize, and tokenize text. This function is used in the machine learning pipeline to vectorize and then apply TF-IDF to the text.

✓

The script builds a pipeline that processes text and then performs multi-output classification on the 36 categories in the dataset. GridSearchCV is used to find the best parameters for the model.

✓

The TF-IDF pipeline is only trained with the training data. The f1 score, precision and recall for the test set is outputted for each category.

Deployment

⟳

The web app, run.py, runs in the terminal without errors. The main page includes at least two visualizations using data from the SQLite database.

The web app, run.py, runs in the terminal without errors. ✅

Required

The main page must include at least two visualizations using data from the SQLite database. KIndly add atleast one more plot.
Just complete the TODO task in the /app/run.py module.

✓

When a user inputs a message into the app, the app returns classification results for all 36 categories.

Classification results are provided based on user inputs.

RESUBMIT

PROJECT LINK

Learn the best practices for revising and resubmitting your project.

RETURN TO PATH

Rate this project

START