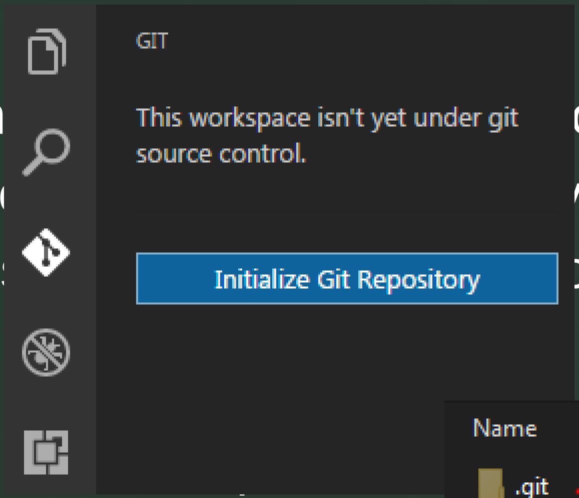
A Recommendation Engine

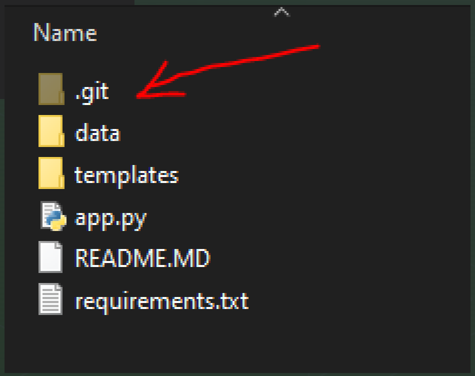
# Kepler API Design

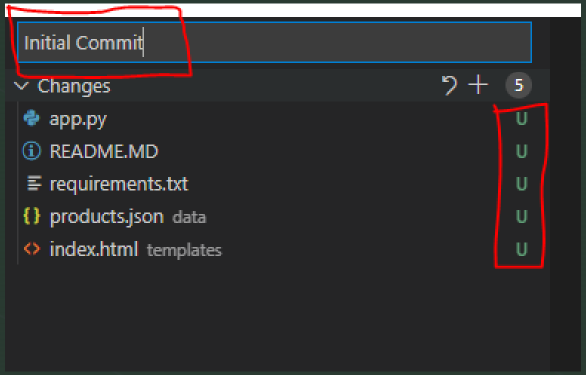
Gene Olafsen

- Start with source files locally
- Click on “Initialize Git Repository” button. This will create a local
  .git folder in the local folder. Th which
  does the magic and keep each ocal
  branches, remote branches, lo and
  every other information which i control
  perspective.
- Once the repository is initialize
  Git has been initialized would start to be tracked.



This workspace isn't yet under git source control.





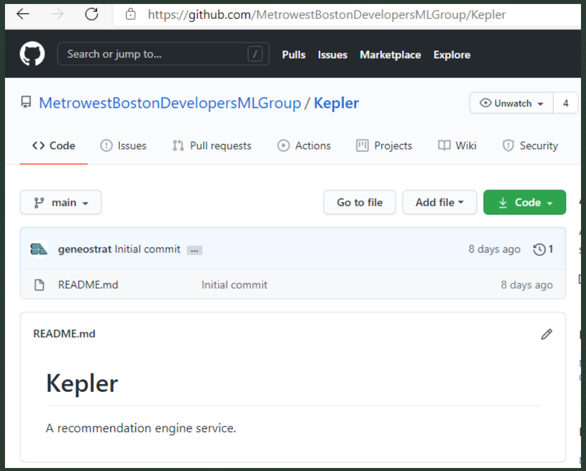
Initial Commit

Changes 5

app.py U
README.MD U
requirements.txt U
products.json data U
index.html templates U

- a

- a


https://github.com/MetrowestBostonDevelopersMLGroup/Kepler

MetrowestBostonDevelopersMLGroup / Kepler

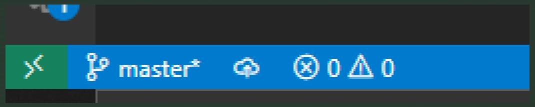
geneostrat Initial commit ... 8 days ago 1

README.md Initial commit 8 days ago

README.md

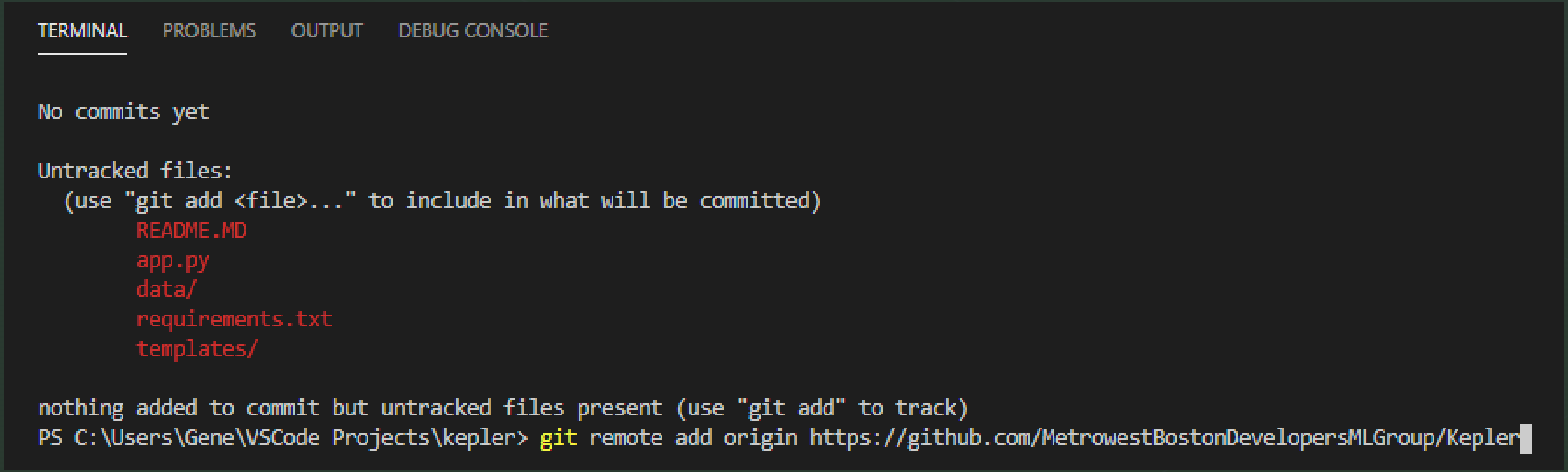
Kepler

A recommendation engine service.



```
PS C:\git\Kepler> git remote add origin https://github.com/MetrowestBostonDevelopersMLGroup/kepler.git
PS C:\git\Kepler> git branch -M main
PS C:\git\Kepler> git push -u origin main
error: src refspec main does not match any
error: failed to push some refs to 'https://github.com/MetrowestBostonDevelopersMLGroup/kepler.git'
PS C:\git\Kepler> git add .
warning: LF will be replaced by CRLF in README.MD.
The file will have its original line endings in your working directory
PS C:\git\Kepler> git commit -m "initial project push"
[main (root-commit) 86a8b0d] initial project push
5 files changed, 661 insertions(+)
create mode 100644 README.MD
create mode 100644 app.py
create mode 100644 data/products.json
create mode 100644 requirements.txt
create mode 100644 templates/index.html
PS C:\git\Kepler> git push -u origin main
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 4 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (9/9), 2.71 KiB | 1.36 MiB/s, done.
Total 9 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/MetrowestBostonDevelopersMLGroup/kepler.git
* [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
PS C:\git\Kepler>
```

- Associate with repo



```
No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        README.MD
        app.py
        data/
        requirements.txt
        templates/

nothing added to commit but untracked files present (use "git add" to track)
PS C:\Users\Gene\VSCode Projects\kepler> git remote add origin https://github.com/MetrowestBostonDevelopersMLGroup/Kepler
```

## Add changes

- Add changes with the checkmark
- Ctrl+Enter
- git push origin -u main

## Load

- Endpoint is provided to load data file(s).
- Currently the service loads the information into memory.
  The Docker image definition can easily be modified to include a SQL or No-SQL database server.

## Describe

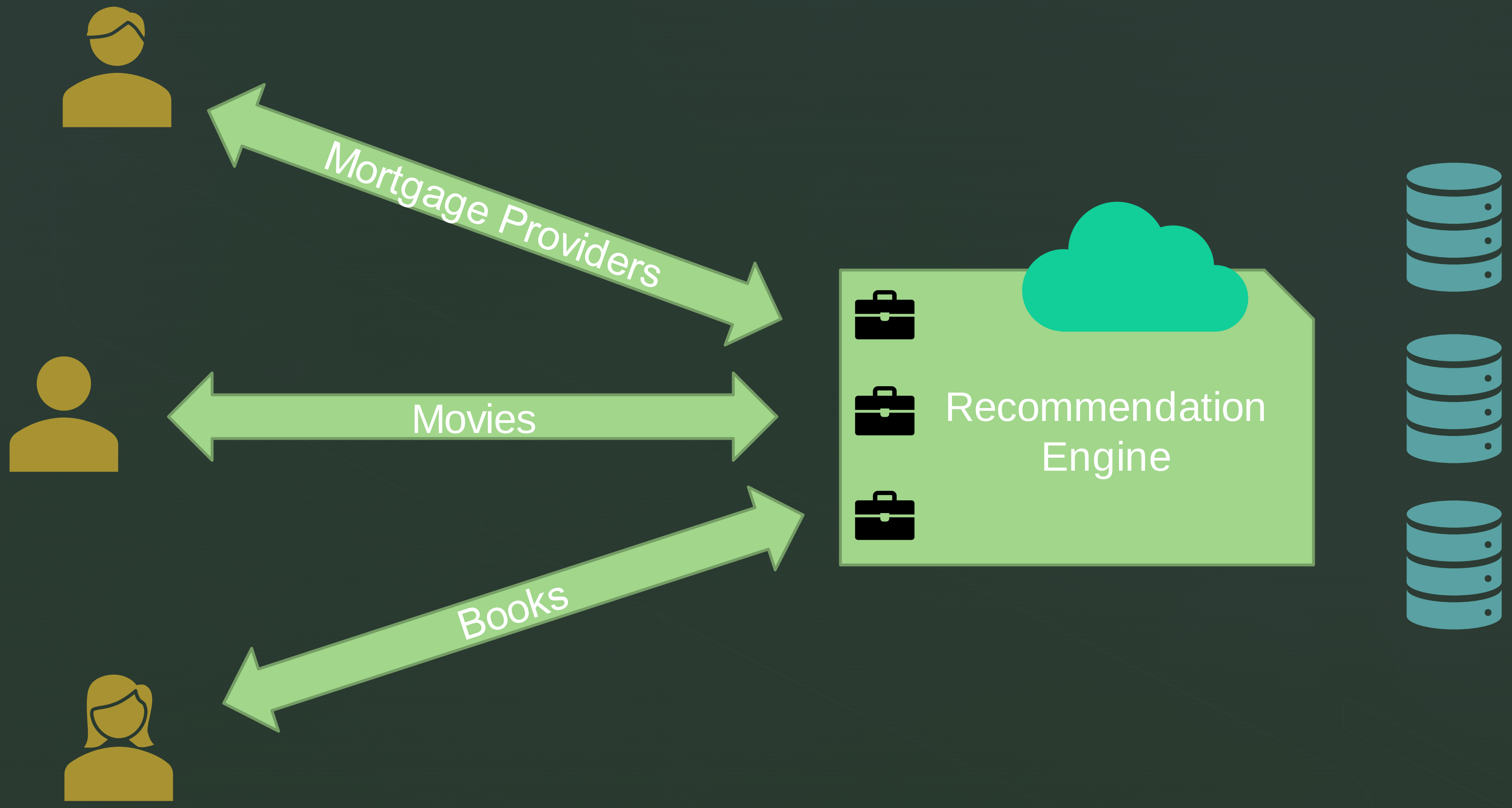
- Source columns must be 'described' so that the recommendation engine can extract data for processing.
- Most columns are 'scaler' types: Int, Bool, Float
- Other data formats are more involved. Consider this JSON field:
  - [{"id": 12, "name": "Adventure"}, {"id": 14, "name": "Fantasy"}, {"id": 28, "name": "Action"}]

## Transform

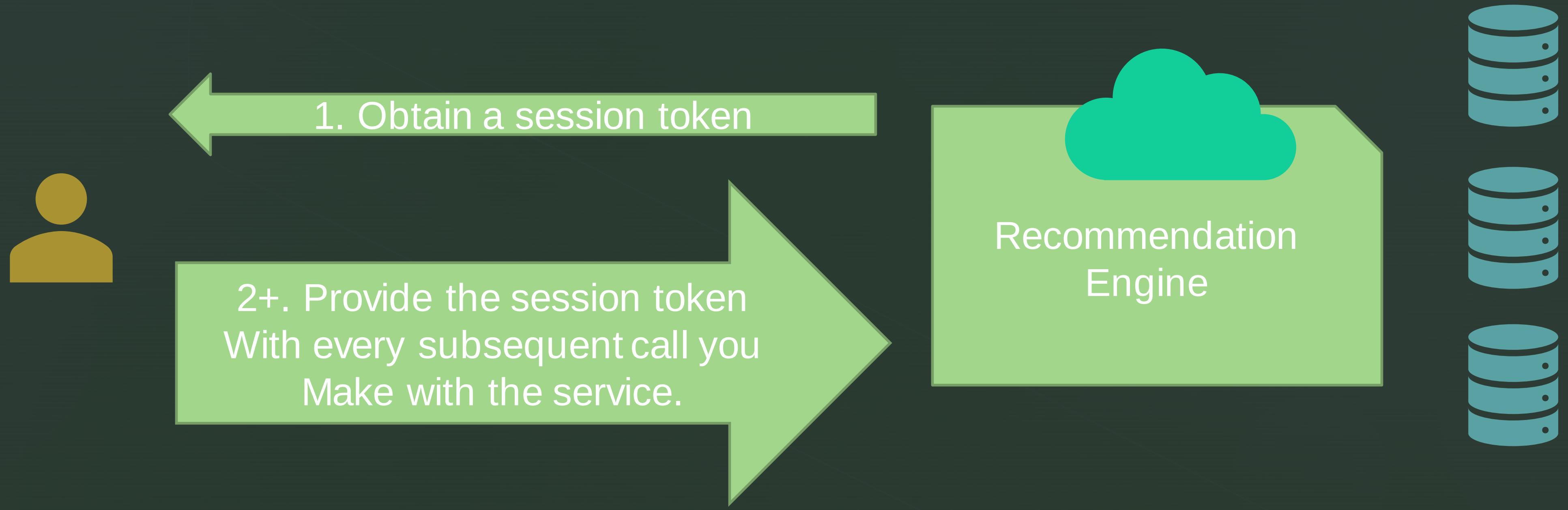
- Transform the information into a format that is suitable for recommendations:
  - Sparse Matrix
  - Bag of Words
  - Tf-Idf Vectorizer (Term Frequency-Inverse Document Frequency)
  - Cosine similarity

## Recommend

## Session State



# Session Conversation


1. Obtain a session token
2+. Provide the session token
With every subsequent call you
Make with the service.
Recommendation
Engine

## Current Endpoints

- @app.route("/loadusmoviesconfig")
  - Executes a recommendation using configuration.
- @app.route("/loadsampleconfig")
  - Executes the parsing of a sample configuration.
- @app.route('/upload', methods=['GET', 'POST'])
  - Uploads a data file to the Docker image

## Audit

http://127.0.0.1:5000/loadusmoviesconfig

```
[
{ "code": 0, "message": "Start audit.", "level": "Info", "extra": "" },
{ "code": 5021, "message": "Working column property is missing the extract-element attribute for this JSON define column.", "level": "Error", "extra": "crew" },
{ "code": 2007, "message": "Working column section JSON column missing item-count attribute. The entire column contents will be extracted by default.", "level": "Warning", "extra": "crew" }
]
```

127.0.0.1:5000/loadsampleconfig

```
 [ { "code": 0, "message": "Start audit.", "level": "Info", "extra": "" },
   { "code": 5000, "message": "This datafile has not been uploaded.", "level": "Error", "extra": "one.csv" },
   { "code": 5014, "message": "Analyze section missing vectorize collection.", "level": "Error", "extra": "" },
   { "code": 5024, "message": "The analyze section is missing the sparse-stack collection: ", "level": "Error", "extra": "" },
   { "code": 5029, "message": "The analyze section is missing the metrics section.", "level": "Error", "extra": "" },
   { "code": 5031, "message": "The configuration file is missing a recommend section.", "level": "Error", "extra": "" },
   { "code": 5000, "message": "This datafile has not been uploaded.", "level": "Error", "extra": "one.csv" } ]
```

## Configuration

- Files
- Transform
- Analyze
- Recommend

## Current Endpoints

- @app.route("/load")
  - Loads and parses the movie db.
- @app.route("/transform")
  - Transforms the movie db.
- @app.route("/recommend")
  - Recommends movies 'like' Alien.
- @app.route("/all")
  - Performs /load, /transform and /recommend - all in one step

## Files Section

- Associate with an uploaded data file.
- Identify 'working columns'
- Describe 'working column' types- JSON
- Specify the columns to:
  - Drop
  - Rename
  - Combine

## na-filter

- I want to focus on one particular argument: `na_filter`, which defaults to `True` and controls how pandas parses null values in the .csv file. Recently, I had been working on some projects where I had it set to `na_filter=False`, which led to unexpected errors in my code. Debugging the issues led to a better understanding of `pd.read_csv` and especially the `na_filter`.

- When you call `pd.read_csv` on a .csv file, the `na_filter` argument looks for values in the file and identifies any that are likely to be NaN, such as blank cells or various iterations of "NA" ("N/A", "NA", "NaN", etc. See the documentation for details). For the corresponding value in the data frame, it places a NaN value.

- This is obviously quite useful, and is the behavior we would typically want.

- Why would we want to set it to `na_filter=False`? One reason is that if we already know our data has no NAs and the file is very large, this method can be faster. Another reason is that you may want to keep strings like "NA" based on your use case. For example, if "NA" is used as shorthand for North America.

- HOWEVER a word of caution: If you use `na_filter=False` and there *are* NA values in your .csv file, they will appear as blank in your dataframe, but will be *invisible to null functions*.

## Files

```
"files" : [
    {
        "filename" : "tmdb_5000_credits.csv",
        "na-filter": true,
        "workingColumns": [
            {
                "header" : "movie_id",
                "is-json" : false
            },
            {
                "header" : "title",
                "is-json" : false
            },
            {
                "header" : "cast",
                "is-json" : true,
                "extract-element"  : "name",
                "item-count": 3
            },
            {
                "header" : "crew",
                "is-json" : true
            }
        ],
        "drop": [],
        "rename" : [{"movie_id": "id"}],
        "combineColumns" : []
    },
```

## Files 2

```
{
    "filename" : "tmdb_5000_movies.csv",
    "na-filter": false,
    "workingColumns": [
        {
            "header" : "id",
            "is-json" : false
        },
        {
            "header" : "genres",
            "is-json" : true,
            "extract-element"  : "name",
            "item-count": 3
        },
        {
            "header" : "overview",
            "is-json" : false
        }
    ],
    "drop": ["tagline", "status", "homepage", "keywords","crew","vote_count",  "vote_average", "tagline", "spoken_languages", "runtime", "popularity", "production_companies", "budget", "production_countries", "release_date", "revenue", "title_y", "original_language"],
    "rename" : [],
    "combineColumns" : [
        {
            "combine-header": "combine",
            "column1" : "cast",
            "column2" : "genres",
            "item-count": 3,
            "drop-source-columns": true
        }
    ]
}
```

The 'Drop' operation may not be necessary as any columns that are not 'working' columns are dropped.

The combine operation is specifying a column named 'cast' that is not part of this data file!

## Transform Overview

- Primarily used (at this point) to merge data files.
- I see this stage providing the ability to perform actions across data files.

## Transform

```
"transform" : {
    "merge": {"from-filename": "tmdb_5000_credits.csv",
    "to-filename":"tmdb_5000_movies.csv",
    "on-column":"id"}
},
```

## Analyze Overview

- This is where the machine learning operations are defined.
- Vectorizers
- Sparse matrix manipulations
- Similarity calculations

## Analyze

```
"analyze" : {
    "vectorizers" : [
        { "id": "countVec", "vectorizer": "count", "stop-words":"english", "column": "combine"},
        { "id": "tfidfVec", "vectorizer": "tfidf", "stop-words":"english", "column": "overview"}
    ],
    "sparse-stack": [
        {
            "id" : "sparse",
            "stack-type" : "hstack",
            "format" : "csr",
            "vectorized-matrix-ids" : ["countVec","tfidfVec"]
        }
    ],
    "metrics": {
        "similarity" : "cosine"
    }
},
```

Intent to support:
API Reference — scikit-learn 0.24.1 documentation (scikit-learn.org)

## Recommend Overview

- Associating a request with a working column.
- Response formatting.

## Recommend

```
"recommend" : {
    "request-column" : "title",
    "response-count" : 20,
    "response-columns" : [
        {"source": "id", "output":"Movie_ID" },
        {"source":"title", "output":"Name"},
        {"source":"genres","output":"Genres"}
    ]
}
```

## Configuration Roadmap

- The analyze section of the configuration may be end up being a 'pipeline' with separate processing stages.
- Data-types may be useful to be defined for each of the working columns.