You:

Write a script in python that takes a txt file as a command line argument that looks like this: INSERT INTO lots_camimage VALUES(2696,'camfeeds/coldwatermi/coldwatermi_202307232226.jpg','2023-07-23 22:26:00','coldwatermi','{"A1": false, "A2": false, "A3": false, "B1": false, "B2": false, "B3": false, "B4": false, "B5": true, "B6": true}','{"A1": false, "A2": false, "A3": false, "B1": false, "B2": false, "B3": false, "B4": false, "B5": true, "B6": true}'); INSERT INTO lots_camimage VALUES(2697,'camfeeds/coldwatermi/coldwatermi_202307232256.jpg','2023-07-23 22:56:00','coldwatermi','{"A1": false, "A2": false, "A3": false, "B1": false, "B2": false, "B3": false, "B4": false, "B5": true, "B6": true}','{"A1": false, "A2": false, "A3": false, "B1": false, "B2": false, "B3": false, "B4": false, "B5": true, "B6": true}'); INSERT INTO lots_camimage VALUES(2698,'camfeeds/coldwatermi/coldwatermi_202307232326.jpg','2023-07-23 23:26:00','coldwatermi','{"A1": false, "A2": false, "A3": false, "B1": false, "B2": false, "B3": false, "B4": false, "B5": true, "B6": true}','{"A1": false, "A2": false, "A3": false, "B1": false, "B2": false, "B3": false, "B4": false, "B5": true, "B6": true}'); and outputs a file that looks like {"coldwatermi_202307232226.jpg": {"A1": false, "A2": false, "A3": false, "B1": false, "B2": false, "B3": false, "B4": false, "B5": true, "B6": true}, "coldwatermi_202307232256.jpg": {"A1": false, "A2": false, "A3": false, "B1": false, "B2": false, "B3": false, "B4": false, "B5": true, "B6": true}, "coldwatermi_202307232326.jpg": {"A1": false, "A2": false, "A3": false, "B1": false, "B2": false, "B3": false, "B4": false, "B5": true, "B6": true}} There will be over 2000 lines

----

ChatGPT:

Sure! I can help you with that.

Here's a script that should accomplish what you're looking for:

```
python
import re
import json
import sys

def convert_sql_to_json(input_file, output_file):
    # Open the input file
    with open(input_file, 'r') as f:
        lines = f.readlines()

    # Create a dictionary to store the result
    result = {}

    # Pattern to extract the filename and the JSON data
    pattern = r"'(camfeeds/coldwatermi/.+?)'.+?'(.+?)'"

    for line in lines:
        matches = re.findall(pattern, line)
        if matches:
            # Extract filename and strip the path
            filename = matches[0][0].split('/')[-1]
            # Extract the JSON data
            data = json.loads(matches[0][1])
            result[filename] = data

    # Save the result to the output file
    with open(output_file, 'w') as f:
        json.dump(result, f, indent=2)
```

```
if __name__ == '__main__':
    if len(sys.argv) != 3:
        print("Usage: script_name.py input_file.txt output_file.json")
        sys.exit(1)

    input_file = sys.argv[1]
    output_file = sys.argv[2]

    convert_sql_to_json(input_file, output_file)
```

To use this script:

1. Save it to a file, e.g., `convert_sql_to_json.py`.
2. Run it using the following command:

```
lua
python convert_sql_to_json.py input.txt output.json
```

Make sure to replace `input.txt` with the name of your input file and `output.json` with the desired name for the output file.

You:

ql_to_json.py", line 37, in <module> convert_sql_to_json(input_file, output_file) File "/home/tom/Desktop/label new pictures/create_correct_base_label/convert_sql_to_json.py", line 22, in convert_sql_to_json data = json.loads(matches[0][1]) File "/home/tom/miniconda3/envs/parkez/lib/python3.9/json/__init__.py", line 346, in loads return _default_decoder.decode(s) File "/home/tom/miniconda3/envs/parkez/lib/python3.9/json/decoder.py", line 340, in decode raise JSONDecodeError("Extra data", s, end) json.decoder.JSONDecodeError: Extra data: line 1 column 5 (char 4)

ChatGPT:

The error you've encountered is a `JSONDecodeError`. This error typically occurs when trying to parse a string as JSON, and the string contains unexpected or excess data.

The error message "Extra data" suggests that there may be some unexpected content in the lines you're trying to parse. The problematic content seems to be starting at column 5, as indicated by "char 4" (since column indexing starts from 0).

The issue could be that the regular expression pattern isn't capturing the data correctly, or there might be some lines in your SQL file that don't match the expected format.

Let's adjust the script to handle unexpected content more gracefully. Here's a modified version:

```
python
import re
import json
import sys
```

```
def convert_sql_to_json(input_file, output_file):
    with open(input_file, 'r') as f:
        lines = f.readlines()

    result = {}
    pattern = r"'(camfeeds/coldwatermi/.+?)'.+?'(.+?)'"

    for line in lines:
        matches = re.findall(pattern, line)
        if matches:
            filename = matches[0][0].split('/')[-1]

            try:
                data = json.loads(matches[0][1])
                result[filename] = data
            except json.JSONDecodeError:
                print(f"Failed to parse JSON for {filename}. Skipping this entry.")

    with open(output_file, 'w') as f:
        json.dump(result, f, indent=2)

if __name__ == '__main__':
    if len(sys.argv) != 3:
        print("Usage: script_name.py input_file.txt output_file.json")
        sys.exit(1)

    input_file = sys.argv[1]
    output_file = sys.argv[2]

    convert_sql_to_json(input_file, output_file)
```

In this version, if there's an error while parsing the JSON for a specific line, it will print a message indicating the problematic file and then skip to the next line. This should help you identify if there are any anomalies in your input file.

You:

It's skipping everything with this input INSERT INTO lots_camimage VALUES(2696,'camfeeds/coldwatermi/coldwatermi_202307232226.jpg','2023-07-23 22:26:00','coldwatermi','{"A1": false, "A2": false, "A3": false, "B1": false, "B2": false, "B3": false, "B4": false, "B5": true, "B6": true}','{"A1": false, "A2": false, "A3": false, "B1": false, "B2": false, "B3": false, "B4": false, "B5": true, "B6": true}'); INSERT INTO lots_camimage VALUES(2697,'camfeeds/coldwatermi/coldwatermi_202307232256.jpg','2023-07-23 22:56:00','coldwatermi','{"A1": false, "A2": false, "A3": false, "B1": false, "B2": false, "B3": false, "B4": false, "B5": true, "B6": true}','{"A1": false, "A2": false, "A3": false, "B1": false, "B2": false, "B3": false, "B4": false, "B5": true, "B6": true}'); INSERT INTO lots_camimage VALUES(2698,'camfeeds/coldwatermi/coldwatermi_202307232326.jpg','2023-07-23 23:26:00','coldwatermi','{"A1": false, "A2": false, "A3": false, "B1": false, "B2": false, "B3": false, "B4": false, "B5": true, "B6": true}','{"A1": false, "A2": false, "A3": false, "B1": false, "B2": false, "B3": false, "B4": false, "B5": true, "B6": true}'); INSERT INTO lots_camimage VALUES(2699,'camfeeds/coldwatermi/coldwatermi_202307232356.jpg','2023-07-23 23:56:00','coldwatermi','{"A1": false, "A2": false, "A3": false, "B1": false, "B2": false, "B3": false, "B4": false, "B5": true, "B6": true}','{"A1": false, "A2": false, "A3": false, "B1": false, "B2": false, "B3": false, "B4": false, "B5": true, "B6": true}'); INSERT INTO lots_camimage VALUES(2700,'camfeeds/coldwatermi/coldwatermi_202307240026.jpg','2023-07-24 00:26:00','coldwatermi','{"A1": false, "A2": false, "A3": false, "B1": false, "B2": false, "B3": false, "B4": false, "B5": true, "B6": true}','{"A1": false, "A2": false, "A3": false, "B1": false, "B2": false, "B3": false, "B4": false, "B5": true, "B6": true}'); INSERT INTO

```
lots_camimage VALUES(2701,'camfeeds/coldwatermi/coldwatermi_202307240057.jpg','2023-07-24
00:57:00','coldwatermi','{"A1": false, "A2": false, "A3": false, "B1": false, "B2": false, "B3": false, "B4": false, "B5":
true, "B6": true}','{"A1": false, "A2": false, "A3": false, "B1": false, "B2": false, "B3": false, "B4": false, "B5": true,
"B6": true}'); INSERT INTO lots_camimage
VALUES(2702,'camfeeds/coldwatermi/coldwatermi_202307240127.jpg','2023-07-24 01:27:00','coldwatermi','{"A1":
false, "A2": false, "A3": false, "B1": false, "B2": false, "B3": false, "B4": false, "B5": true, "B6": true}','{"A1": false,
"A2": false, "A3": false, "B1": false, "B2": false, "B3": false, "B4": false, "B5": true, "B6": true}'); INSERT INTO
lots_camimage VALUES(2703,'camfeeds/coldwatermi/coldwatermi_202307240157.jpg','2023-07-24
01:57:00','coldwatermi','{"A1": false, "A2": false, "A3": false, "B1": false, "B2": false, "B3": false, "B4": false, "B5":
true, "B6": true}','{"A1": false, "A2": false, "A3": false, "B1": false, "B2": false, "B3": false, "B4": false, "B5": true,
"B6": true}'); INSERT INTO lots_camimage
VALUES(2704,'camfeeds/coldwatermi/coldwatermi_202307240227.jpg','2023-07-24 02:27:00','coldwatermi','{"A1":
false, "A2": false, "A3": false, "B1": false, "B2": false, "B3": false, "B4": false, "B5": true, "B6": true}','{"A1": false,
"A2": false, "A3": false, "B1": false, "B2": false, "B3": false, "B4": false, "B5": true, "B6": true}'); INSERT INTO
lots_camimage VALUES(2705,'camfeeds/coldwatermi/coldwatermi_202307240257.jpg','2023-07-24
02:57:00','coldwatermi','{"A1": false, "A2": false, "A3": false, "B1": false, "B2": false, "B3": false, "B4": false, "B5":
true, "B6": true}','{"A1": false, "A2": false, "A3": false, "B1": false, "B2": false, "B3": false, "B4": false, "B5": true,
"B6": true}'); INSERT INTO lots_camimage
VALUES(2706,'camfeeds/coldwatermi/coldwatermi_202307240327.jpg','2023-07-24 03:27:00','coldwatermi','{"A1":
false, "A2": false, "A3": false, "B1": false, "B2": false, "B3": false, "B4": false, "B5": true, "B6": true}','{"A1": false,
"A2": false, "A3": false, "B1": false, "B2": false, "B3": false, "B4": false, "B5": true, "B6": true}'); INSERT INTO
lots_camimage VALUES(2707,'camfeeds/coldwatermi/coldwatermi_202307240358.jpg','2023-07-24
03:58:00','coldwatermi','{"A1": false, "A2": false, "A3": false, "B1": false, "B2": false, "B3": false, "B4": false, "B5":
true, "B6": true}','{"A1": false, "A2": false, "A3": false, "B1": false, "B2": false, "B3": false, "B4": false, "B5": true,
"B6": true}'); INSERT INTO lots_camimage
VALUES(2708,'camfeeds/coldwatermi/coldwatermi_202307240428.jpg','2023-07-24 04:28:00','coldwatermi','{"A1":
false, "A2": false, "A3": false, "B1": false, "B2": false, "B3": false, "B4": false, "B5": true, "B6": true}','{"A1": false,
"A2": false, "A3": false, "B1": false, "B2": false, "B3": false, "B4": false, "B5": true, "B6": true}');
```

ChatGPT:

I see the issue. There are two JSON objects in each line of the input, but the regex pattern is capturing only one. Let's modify the regex pattern to capture the correct JSON object, which seems to be the fifth column in the input.

Here's the modified version of the script that should handle your input correctly:

```
python
import re
import json
import sys

def convert_sql_to_json(input_file, output_file):
    with open(input_file, 'r') as f:
        lines = f.readlines()

    result = {}
    # Adjusting the pattern to capture the correct JSON data
    pattern = r"'(camfeeds/coldwatermi/.+?)'.+?'.+?'.+?','(.+?)','"

    for line in lines:
        matches = re.findall(pattern, line)
        if matches:
            filename = matches[0][0].split('/')[-1]

            try:
                data = json.loads(matches[0][1])
                result[filename] = data
            except json.JSONDecodeError:
                print(f"Failed to parse JSON for {filename}. Skipping this entry.")

    with open(output_file, 'w') as f:
        json.dump(result, f, indent=2)

if __name__ == '__main__':
    if len(sys.argv) != 3:
        print("Usage: script_name.py input_file.txt output_file.json")
        sys.exit(1)

    input_file = sys.argv[1]
    output_file = sys.argv[2]

    convert_sql_to_json(input_file, output_file)
```

This adjustment in the regex pattern will ensure that the script captures the correct JSON object from the fifth column in the input.

You:

How do a load a file in json format into a dataframe in jupyter notebook

----

ChatGPT:

To load a JSON file into a DataFrame in a Jupyter Notebook, you can use the `pandas` library. If you don't already have it installed, you can install it using `pip`:

```
diff
!pip install pandas
```

Once you have `pandas` installed, you can use the following steps to load a JSON file into a DataFrame:

1. Import the necessary libraries.
2. Use the `read_json` function from `pandas` to read the JSON file and load it into a DataFrame.

Here's a code snippet to demonstrate:

```
python
import pandas as pd

# Load JSON file into a DataFrame
df = pd.read_json('/path/to/your/file.json')

# Display the first few rows of the DataFrame
df.head()
```

Replace `'/path/to/your/file.json'` with the path to your JSON file. After executing these steps, the variable `df` will contain the contents of the JSON file as a DataFrame, and you can perform any DataFrame operations on it as you would normally do.

You:

How you read it with reversed columns and rows?

ChatGPT:

To read a JSON file into a DataFrame with reversed columns and rows (i.e., transposed), you can simply use the `transpose()` method of the DataFrame after reading the JSON file. Here's how you can do it:

```
python
import pandas as pd

# Load JSON file into a DataFrame
df = pd.read_json('/path/to/your/file.json')

# Transpose the DataFrame
df_transposed = df.transpose()

# Display the first few rows of the transposed DataFrame
df_transposed.head()
```

Replace `'/path/to/your/file.json'` with the path to your JSON file. After executing these steps, the variable `df_transposed` will contain the transposed version of the data from the JSON file.

You:

how do I get all row names?

ChatGPT:

To get all the row names (also known as the index) of a DataFrame in pandas, you can use the `.index` attribute.

Here's how you can retrieve and display the row names (index) of the DataFrame:

```
python
row_names = df_transposed.index.tolist()
print(row_names)
```

Executing this code will display a list of all the row names in the `df_transposed` DataFrame. If you're working with the original `df` DataFrame, simply replace `df_transposed` with `df` in the code.

You:

How do you make these lists: files_list[171:192] files_list[484] files_list[492:506] files_list[826:839] files_list[921:936] files_list[1135:1147] files_list[1795:1807] One list and then print each list item with a space between in python

----

ChatGPT:

Here's the combined list with each item separated by a space:

```
file171 file172 file173 file174 file175 ... file1804 file1805 file1806
```

(Note: This is just a portion of the output for clarity; the full list has been generated above.)

You can use the given output directly or apply any further processing you require.

You:

Whats to easiest way to load a json file that looks like this in python: {"coldwatermi_202307232226.jpg": {"A1": false, "A2": false, "A3": false, "B1": false, "B2": false, "B3": false, "B4": false, "B5": true, "B6": true}, "coldwatermi_202307232256.jpg": {"A1": false, "A2": false, "A3": false, "B1": false, "B2": false, "B3": false, "B4": false, "B5": true, "B6": true}, "coldwatermi_202307232326.jpg": {"A1": false, "A2": false, "A3": false, "B1": false, "B2": false, "B3": false, "B4": false, "B5": true, "B6": true}, "coldwatermi_202307232356.jpg": {"A1": false, "A2": false, "A3": false, "B1": false, "B2": false, "B3": false, "B4": false, "B5": false, "B6": false}, "coldwatermi_202307240026.jpg": {"A1": false, "A2": false, "A3": false, "B1": false, "B2": false, "B3": false, "B4": false, "B5": false, "B6": false}, "coldwatermi_202307240057.jpg": {"A1": false, "A2": false, "A3": false, "B1": false, "B2": false, "B3": false, "B4": false, "B5": false, "B6": false}, "coldwatermi_202307240127.jpg": {"A1": false, "A2": false, "A3": false, "B1": false, "B2": false, "B3": false, "B4": false, "B5": false, "B6": false}, "coldwatermi_202307240157.jpg": {"A1": false, "A2": false, "A3": false, "B1": false, "B2": false, "B3": false, "B4": false, "B5": false, "B6": false}, "coldwatermi_202307240227.jpg": {"A1": false, "A2": false, "A3": false, "B1": false, "B2": false, "B3": false, "B4": false, "B5": false, "B6": false}, "coldwatermi_202307240257.jpg": {"A1": false, "A2": false, "A3": false, "B1": false, "B2": false, "B3": false, "B4": false, "B5": false, "B6": false}, "coldwatermi_202307240327.jpg": {"A1": false, "A2": false, "A3": false, "B1": false, "B2": false, "B3": false, "B4": false, "B5": false, "B6": false}, "coldwatermi_202307240358.jpg": {"A1": false, "A2": false, "A3": false, "B1": false, "B2": false, "B3": false, "B4": false, "B5": false, "B6": false}, "coldwatermi_202307240428.jpg": {"A1": false, "A2": false, "A3": false, "B1": false, "B2": false, "B3": false, "B4": false, "B5": false, "B6": false}, "coldwatermi_202307240458.jpg": {"A1": false, "A2": false, "A3": false, "B1": false, "B2": false, "B3": false, "B4": false, "B5": false, "B6": false}, "coldwatermi_202307240528.jpg": {"A1": false, "A2": false, "A3": false, "B1": false, "B2": false, "B3": false, "B4": false, "B5": false, "B6": false}, "coldwatermi_202307240558.jpg": {"A1": false, "A2": false, "A3": false, "B1": false, "B2": false, "B3": false, "B4": false, "B5": false, "B6": false}} Then remove a series of entries with these keys for example: coldwatermi_202307240558.jpg coldwatermi_202307240127.jpg coldwatermi_202307232256.jpg And then save the a new JSON file with the entries with the above keys removed, but otherwise preserve the order of the keys in the JSON file (which may happen without doing naything)

ChatGPT:

The specified entries have been removed from the JSON data, and the modified data has been saved to a new JSON file named `modified_data.json`.

You can now access the `modified_data.json` file for the updated data. If you're running this in a local Jupyter notebook environment, you'll find the file in the current working directory. If you're using this platform, you can download the file directly from the provided path.

You:

Just one addtional step: How do I load my .json file

ChatGPT:

The JSON file `modified_data.json` has been successfully loaded into the Python dictionary named `loaded_data`. As a verification step, I've displayed the first item from the loaded data.

Now, you can use the `loaded_data` dictionary in your Python code to access or manipulate the JSON data as needed.

You:

import os import json

ChatGPT:

It seems you want to import the `os` and `json` modules in Python. These are standard libraries in Python and can be imported directly.

Here's how you do it:

```
python
import os
import json
```

Once imported, you can use the functions and classes provided by these modules. For instance, `os` gives you a way of using operating system dependent functionality, such as reading or writing to the file system, while `json` allows you to encode and decode JSON data.

Is there a specific task you'd like to accomplish using these modules?

You:

import os import json with open('human.json', 'r') as json_file: human_json = json.load(json_file) with open('machine.json', 'r') as json_file: machine_json = json.load(json_file) for x in list(machine_json.keys()): if x not in list(human_json.keys()): print(x) Make a list of all the entries that are both in list(machine_json.keys()) and list(machine_json.keys())

You:

import os import json with open('human.json', 'r') as json_file: human_json = json.load(json_file) with open('machine.json', 'r') as json_file: machine_json = json.load(json_file) for x in list(machine_json.keys()): if x not in list(human_json.keys()): print(x) Make a list of all the entries that are both in list(machine_json.keys()) and list(machine_json.keys())

ChatGPT:

It seems there was an issue while processing the request. Let's break down the task again:

1. Load the `human.json` and `machine.json` files.
2. Find the keys that are common to both files.
3. Create a list of these common keys.

I'll write the code to accomplish this for you:

```
python
import os
import json

# Load the human.json and machine.json files
with open('human.json', 'r') as json_file:
    human_json = json.load(json_file)

with open('machine.json', 'r') as json_file:
    machine_json = json.load(json_file)

# Create a list of keys that are common to both machine_json and human_json
common_keys = [key for key in machine_json.keys() if key in human_json.keys()]

common_keys
```

This will give you a list of keys that are present in both the `machine_json` and `human_json` files.

You:

This code: for x in common_keys[10:14]: print('\n' + x) print("human label") print(human_json[x]) print("machine prediction") print(machine_json[x]) Produces this output: coldwatermi_202307240327.jpg human label {'A1': False, 'A2': False, 'A3': False, 'B1': False, 'B2': False, 'B3': False, 'B4': False, 'B5': True, 'B6': True} machine prediction {'A1': False, 'A2': False, 'A3': False, 'B1': False, 'B2': False, 'B3': False, 'B4': False, 'B5': True, 'B6': True} coldwatermi_202307240358.jpg human label {'A1': False, 'A2': False, 'A3': False, 'B1': False, 'B2': False, 'B3': False, 'B4': False, 'B5': True, 'B6': True} machine prediction {'A1': False, 'A2': False, 'A3': False, 'B1': False, 'B2': False, 'B3': False, 'B4': False, 'B5': True, 'B6': True} coldwatermi_202307240428.jpg human label {'A1': False, 'A2': False, 'A3': False, 'B1': False, 'B2': False, 'B3': False, 'B4': False, 'B5': True, 'B6': True} machine prediction {'A1': False, 'A2': False, 'A3': False, 'B1': False, 'B2': False, 'B3': False, 'B4': False, 'B5': True, 'B6': True} coldwatermi_202307240458.jpg human label {'A1': False, 'A2': False, 'A3': False, 'B1': False, 'B2': False, 'B3': False, 'B4': False, 'B5': True, 'B6': True} machine prediction {'A1': False, 'A2': False, 'A3': False, 'B1': False, 'B2': False, 'B3': False, 'B4': False, 'B5': True, 'B6': True} Modify this code so it adds up true positives, false positives, true negatives and false negatives, both for A1, A2, A3, B1, B2, B3, B4, B5 and B6 and for all combined.

ChatGPT:

I apologize for the oversight. Let me first initialize the necessary data structures and then proceed with the calculations. I'll use the provided sample JSON data to simulate the operations.

You:

Given this result for a expanded version of the code above (but on many more common keys) compute the traditional ML metrics for evaluating a binary classifier for A1-B6 and for all combined: {'A1': {'TP': 473, 'FP': 2, 'TN': 1702, 'FN': 11}, 'A2': {'TP': 594, 'FP': 5, 'TN': 1583, 'FN': 6}, 'A3': {'TP': 611, 'FP': 18, 'TN': 1554, 'FN': 5}, 'B1': {'TP': 384, 'FP': 3, 'TN': 1800, 'FN': 1}, 'B2': {'TP': 480, 'FP': 0, 'TN': 1707, 'FN': 1}, 'B3': {'TP': 585, 'FP': 1, 'TN': 1602, 'FN': 0}, 'B4': {'TP': 559, 'FP': 4, 'TN': 1624, 'FN': 1}, 'B5': {'TP': 670, 'FP': 26, 'TN': 1488, 'FN': 4}, 'B6': {'TP': 589, 'FP': 33, 'TN': 1501, 'FN': 65}, 'combined': {'TP': 4945, 'FP': 92, 'TN': 14561, 'FN': 94}}

ChatGPT:

Here are the traditional ML metrics for evaluating a binary classifier for each label and the combined metrics:

A1:

- Accuracy: 99.41%99.41%
- Precision: 99.58%99.58%
- Recall: 97.73%97.73%
- Specificity: 99.88%99.88%
- F1 Score: 98.64%98.64%

A2:

- Accuracy: 99.50%99.50%
- Precision: 99.17%99.17%
- Recall: 99.00%99.00%

- Specificity: 99.69%99.69%
- F1 Score: 99.08%99.08%

A3:

- Accuracy: 98.95%98.95%
- Precision: 97.14%97.14%
- Recall: 99.19%99.19%
- Specificity: 98.85%98.85%
- F1 Score: 98.15%98.15%

B1:

- Accuracy: 99.82%99.82%
- Precision: 99.22%99.22%
- Recall: 99.74%99.74%
- Specificity: 99.83%99.83%
- F1 Score: 99.48%99.48%

B2:

- Accuracy: 99.95%99.95%
- Precision: 100.00%100.00%
- Recall: 99.79%99.79%
- Specificity: 100.00%100.00%
- F1 Score: 99.90%99.90%

B3:

- Accuracy: 99.95%99.95%
- Precision: 99.83%99.83%
- Recall: 100.00%100.00%
- Specificity: 99.94%99.94%
- F1 Score: 99.91%99.91%

B4:

- Accuracy: 99.77%99.77%
- Precision: 99.29%99.29%
- Recall: 99.82%99.82%
- Specificity: 99.75%99.75%
- F1 Score: 99.55%99.55%

B5:

- Accuracy: 98.63%98.63%
- Precision: 96.26%96.26%
- Recall: 99.41%99.41%
- Specificity: 98.28%98.28%
- F1 Score: 97.81%97.81%

B6:

- Accuracy: 95.52%95.52%
- Precision: 94.69%94.69%
- Recall: 90.06%90.06%
- Specificity: 97.85%97.85%
- F1 Score: 92.32%92.32%

Combined:

- Accuracy: 99.06%99.06%
- Precision: 98.17%98.17%
- Recall: 98.13%98.13%
- Specificity: 99.37%99.37%
- F1 Score: 98.15%98.15%

These metrics will help you evaluate the performance of your binary classifier against the human labels.

You:

How do I take this dict in python: {'A1': {'Accuracy': 0.9940585009140768, 'Precision': 0.9957894736842106, 'Recall': 0.9772727272727273, 'Specificity': 0.9988262910798122, 'F1 Score': 0.986444212721585}, 'A2': {'Accuracy': 0.9949725776965265, 'Precision': 0.991652754590985, 'Recall': 0.99, 'Specificity': 0.9968513853904282, 'F1 Score': 0.9908256880733944}, 'A3': {'Accuracy': 0.9894881170018281, 'Precision': 0.9713831478537361, 'Recall': 0.9918831168831169, 'Specificity': 0.9885496183206107, 'F1 Score': 0.9815261044176707}, 'B1': {'Accuracy': 0.9981718464351006, 'Precision': 0.9922480620155039, 'Recall': 0.9974025974025974, 'Specificity': 0.9983361064891847, 'F1 Score': 0.994818652849741}, 'B2': {'Accuracy': 0.9995429616087751, 'Precision': 1.0, 'Recall': 0.997920997920998, 'Specificity': 1.0, 'F1 Score': 0.9989594172736733}, 'B3': {'Accuracy': 0.9995429616087751, 'Precision': 0.9982935153583617, 'Recall': 1.0, 'Specificity': 0.9993761696818465, 'F1 Score': 0.9991460290350128}, 'B4': {'Accuracy': 0.9977148080438757, 'Precision': 0.9928952042628775, 'Recall': 0.9982142857142857, 'Specificity': 0.9975429975429976, 'F1 Score': 0.9955476402493322}, 'B5': {'Accuracy': 0.9862888482632541, 'Precision': 0.9626436781609196, 'Recall': 0.9940652818991098, 'Specificity': 0.9828269484808454, 'F1 Score': 0.9781021897810219}, 'B6': {'Accuracy': 0.9552102376599635, 'Precision': 0.9469453376205788, 'Recall': 0.900611620795107, 'Specificity': 0.9784876140808344, 'F1 Score': 0.9231974921630094}, 'combined': {'Accuracy': 0.9905545399146861, 'Precision': 0.9817351598173516, 'Recall': 0.9813455050605279, 'Specificity': 0.9937214222343548, 'F1 Score': 0.981540293767368}} And transform it to a Github .md table that looks like ||A1|A2|A3|B1|B2|B3|B4|B5|B6|all |Accuracy|10.0%|10.0%|10.0%|10.0%|10.0%|10.0%|10.0%|10.0%|10.0%|10.0% |Recall|10.0%|10.0%|10.0%|10.0%|10.0%|10.0%|10.0%|10.0%|10.0%|10.0% |Specificity|10.0%|10.0%|10.0%|10.0%|10.0%|10.0%|10.0%|10.0%|10.0%|10.0% |F1 Score|10.0%|10.0%|10.0%|10.0%|10.0%|10.0%|10.0%|10.0%|10.0%|10.0% But with real values

----

ChatGPT:

Certainly! Here's the transformation of the dictionary to a Github markdown table:

```
markdown
||A1|A2|A3|B1|B2|B3|B4|B5|B6|All
|---|---|---|---|---|---|---|---|---|---|---|
|Accuracy|99.41%|99.50%|98.95%|99.82%|99.95%|99.95%|99.77%|98.63%|95.52%|99.06%|
|Precision|99.58%|99.17%|97.14%|99.22%|100.00%|99.83%|99.29%|96.26%|94.69%|98.17%|
|Recall|97.73%|99.00%|99.19%|99.74%|99.79%|100.00%|99.82%|99.41%|90.06%|98.13%|
|Specificity|99.88%|99.69%|98.85%|99.83%|100.00%|99.94%|99.75%|98.28%|97.85%|99.37%|
|F1 Score|98.64%|99.08%|98.15%|99.48%|99.90%|99.91%|99.55%|97.81%|92.32%|98.15%|
```

Copy and paste the markdown code above into your Github markdown file to get the desired table.
