

Question

What is the age of the patient with the lowest APP-Z score?

Ground Truth Files

1-s2.0-S0092867420301070-mmc1.xlsx
1-s2.0-S0092867420301070-mmc7.xlsx

Ground Truth Response

60

Master-Slave

Generated Response: 74

```
import pandas as pd
import json

def get_age_and_app_scores(files):
    """
    Loads patient age and APP protein abundance scores from the supplementary files.

    Args:
        files (list): A list of all file paths in the dataset.

    Returns:
        pandas.DataFrame: A DataFrame with 'Age' and 'APP_log2_abundance' for each sample.
    """

# Find the metadata and proteomics files from the list
meta_file = next((f for f in files if 'mmc1.xlsx' in f), None)
prot_file = next((f for f in files if 'mmc2.xlsx' in f), None)
```

```
if not meta_file or not prot_file:
    raise FileNotFoundError("Required metadata (mmc1) or proteomics (mmc2) file not found.")
```

```
# Load the clinical metadata to get patient age
df_meta = pd.read_excel(meta_file, sheet_name='UCEC_CPTAC3_meta_table_V2.1', index_col=0)
age_data = df_meta[['Age']]
```

```
# Load the global proteomics data to get protein scores
df_prot = pd.read_excel(prot_file, sheet_name='A-global-proteomics', index_col=0)
```

```
# Check if the 'APP' protein is in the proteomics data
if 'APP' in df_prot.index:
    # Extract the row for 'APP', transpose it to a column, and rename it
    app_scores = df_prot.loc[['APP']].T
    app_scores.rename(columns={'APP': 'APP_log2_abundance'}, inplace=True)
else:
    # If APP is not found, create a DataFrame with NaNs to allow merging
    app_scores = pd.DataFrame(index=df_meta.index, columns=['APP_log2_abundance'])
```

```
# Combine the age and APP score dataframes
combined_df = age_data.join(app_scores)
```

```
# Remove samples that are missing either Age or the APP score
combined_df.dropna(subset=['Age', 'APP_log2_abundance'], inplace=True)
```

```
return combined_df
```

```
# List of all available files
```

```
file_list = [
    '1-s2.0-S0092867420301070-mmc1.xlsx',
    '1-s2.0-S0092867420301070-mmc2.xlsx',
    '1-s2.0-S0092867420301070-mmc3.xlsx',
    '1-s2.0-S0092867420301070-mmc4.xlsx',
    '1-s2.0-S0092867420301070-mmc5.xlsx',
    '1-s2.0-S0092867420301070-mmc6.xlsx',
    '1-s2.0-S0092867420301070-mmc7.xlsx'
]
```

```
# Execute the function to get the data
```

```
age_app_data = get_age_and_app_scores(file_list)
```

```
# Find the patient with the lowest APP-Z score
```

```
patient_with_lowest_app = age_app_data.loc[age_app_data['APP_log2_abundance'].idxmin()]
```

```
# Get the age of that patient
```

```
age = patient_with_lowest_app['Age']
```

```
# Create the final answer dictionary
```

```
final_answer = {
    "main-task": age
}
```

```
# Print the final answer in JSON format
print(json.dumps(final_answer, indent=4))
```

Blackboard

Generated Response: 60

```
import pandas as pd
import json

# Define file paths
file_path_meta = '1-s2.0-S0092867420301070-mmc1.xlsx'
file_path_apm = '1-s2.0-S0092867420301070-mmc7.xlsx'

# --- Solution for Subtask 1: Load and merge data ---
# Load the main metadata sheet from mmc1.xlsx which contains patient 'Age'.
meta_df = pd.read_excel(file_path_meta, sheet_name='UCEC_CPTAC3_meta_table_V2.1', index_col=0)

# Load the APM subtypes sheet from mmc7.xlsx which contains the 'APP_Z_score'.
apm_df = pd.read_excel(file_path_apm, sheet_name='B-APM subtypes', index_col=0)

# Select only the necessary columns
patient_info_df = meta_df[['Age']]
apm_scores_df = apm_df[['APP_Z_score']]

# Merge the two dataframes based on the sample ID index
merged_df = patient_info_df.join(apm_scores_df, how='inner')
subtask1_answer = "Data loaded and merged successfully. The merged data contains 'Age' and 'APP_Z_score' for each patient."

# --- Solution for Subtask 2: Clean the data ---
# Convert columns to numeric, coercing errors to NaN
merged_df['Age'] = pd.to_numeric(merged_df['Age'], errors='coerce')
merged_df['APP_Z_score'] = pd.to_numeric(merged_df['APP_Z_score'], errors='coerce')

# Drop rows with missing values in 'Age' or 'APP_Z_score'
cleaned_df = merged_df.dropna(subset=['Age', 'APP_Z_score'])
subtask2_answer = "Data cleaned. Rows with missing 'Age' or 'APP_Z_score' have been removed."

# --- Solution for Main Task: Find the age of the patient with the lowest APP-Z score ---
# Find the index of the patient with the lowest APP-Z score
patient_with_lowest_score_idx = cleaned_df['APP_Z_score'].idxmin()

# Get the age of that patient
age_of_patient = cleaned_df.loc[patient_with_lowest_score_idx, 'Age']

# The final answer is the age of the patient, converted to an integer.
final_answer = int(age_of_patient)

# Print the final results in JSON format
print(json.dumps({
    "subtask-1": subtask1_answer,
    "subtask-2": subtask2_answer,
    "main-task": final_answer
}, indent=4))
```