```python
import sys
sys.path.append('C:/Program Files/Stata17/utilities')
from pystata import config
# Try a different edition if you're not sure which one you have
config.init('se')  # Try Stata/SE instead of MP
# Or
config.init('be')  # Try Stata/BE

import tempfile
import os
import pandas as pd
from pystata import stata

# Step 1: Load the Stata dataset via pystata
stata.run('use "Finalized_PSLM_Sheikhupura.dta", clear')

# Optionally, run any preprocessing commands in Stata here:
# stata.run('drop if missing(some_variable)')
# stata.run('keep if condition')

# Step 2: Create a temporary CSV file for exporting data
temp_csv = tempfile.NamedTemporaryFile(delete=False, suffix=".csv")
temp_csv.close()  # Close the file so that Stata can write to it

# Export the dataset to the temporary CSV file using Stata's export
command
stata.run(f'export delimited using "{temp_csv.name}", replace')

# Step 3: Read the exported CSV into a Pandas DataFrame
df = pd.read_csv(temp_csv.name,low_memory=False)

# (Optional) Remove the temporary file now that the data is loaded
os.unlink(temp_csv.name)

# Verify the data loaded correctly
print(df.head())
```

```
file C:\Users\PMLS\AppData\Local\Temp\tmpkgx8_0o2.csv saved
      hhcode      psu province region     district  idc
relationship_to_head  \
0  234100103  2341001   punjab  rural  sheikhupura    3
son/daughter
1  234100103  2341001   punjab  rural  sheikhupura    4
son/daughter
2  234100104  2341001   punjab  rural  sheikhupura    6
son/daughter
3  234100104  2341001   punjab  rural  sheikhupura    5
son/daughter
4  234100104  2341001   punjab  rural  sheikhupura    4
son/daughter
```

```
      reason_for_headship  gender residence_status  ...  annual_income
\
0  main economic provider  female          present  ...            NaN

1  main economic provider    male          present  ...            NaN

2  main economic provider  female          present  ...            NaN

3  main economic provider  female          present  ...            NaN

4  main economic provider    male          present  ...            NaN


   financial_assistance  receives_assistance  income_category
edu_level  \
0                     0        No Assistance       Low Income
NaN
1                     0        No Assistance       Low Income
NaN
2                     0        No Assistance       Low Income
NaN
3                     0        No Assistance       Low Income
NaN
4                     0        No Assistance       Low Income
NaN

   edu_level_num  head_gender  employment_income
total_household_income  \
0            NaN            1                NaN
NaN
1            NaN            1                NaN
NaN
2            NaN            1                NaN
NaN
3            NaN            1                NaN
NaN
4            NaN            1                NaN
NaN

  disability
0         No
1         No
2         No
3         No
4         No

[5 rows x 381 columns]
```

```
# categorical_features = df.select_dtypes(include=['object',
'category']).columns
# print("Number of categorical features:", len(categorical_features))
# print("Categorical features:")
# print(categorical_features.tolist())

features = [
    'age', 'birth_last_3yrs', 'born_district_type',
'born_in_district', 'can_do_math',
    'can_read', 'can_report_income', 'can_write', 'computer_location',
'connected_to_sewerage',
    'cooking_water_source', 'disability', 'dwelling_type',
'employment_status',
    'first_prenatal_visit_month', 'gender',
'handwashing_water_source', 'has_computer',
    'has_handwashing_place', 'has_internet', 'has_job', 'has_mobile',
'has_mobile_phone',
    'has_property','total_household_income',
    'house_owner_gender', 'household_member',
'household_ran_out_of_food',
    'hungry_but_did_not_eat', 'income_used_for_hh','marital_status',
    'migration_reason', 'monthly_income', 'no_computer_reason',
'no_mobile_reason',
    'num_prenatal_visits', 'num_rooms', 'pay_for_water',
'prenatal_consultation_source',
    'prenatal_consultations', 'property_owner_gender',
'reason_for_headship', 'region',
    'relationship_to_head', 'residence_status', 'sick_last_2wks',
'shared_toilet',
    'sufficient_drinking_water', 'toilet_type', 'used_computer',
'used_mobile',
    'work_days_last_month', 'worked_last_month', 'worried_about_food',
'disability', 'has_mobile_phone',

'relationship_to_head','house_owner_gender','years_to_complete_primary
'


]

 target = 'edu_access'
 # Separate features and target variable
 X = df[features]
 y = df[target]

column_list = X.columns.tolist()
print(column_list)
```

```
['age', 'birth_last_3yrs', 'born_district_type', 'born_in_district',
'can_do_math', 'can_read', 'can_report_income', 'can_write',
'computer_location', 'connected_to_sewerage', 'cooking_water_source',
```

```
'disability', 'dwelling_type', 'employment_status',
'first_prenatal_visit_month', 'gender', 'handwashing_water_source',
'has_computer', 'has_handwashing_place', 'has_internet', 'has_job',
'has_mobile', 'has_mobile_phone', 'has_property',
'total_household_income', 'house_owner_gender', 'household_member',
'household_ran_out_of_food', 'hungry_but_did_not_eat',
'income_used_for_hh', 'marital_status', 'migration_reason',
'monthly_income', 'no_computer_reason', 'no_mobile_reason',
'num_prenatal_visits', 'num_rooms', 'pay_for_water',
'prenatal_consultation_source', 'prenatal_consultations',
'property_owner_gender', 'reason_for_headship', 'region',
'relationship_to_head', 'residence_status', 'sick_last_2wks',
'shared_toilet', 'sufficient_drinking_water', 'toilet_type',
'used_computer', 'used_mobile', 'work_days_last_month',
'worked_last_month', 'worried_about_food', 'disability',
'has_mobile_phone', 'relationship_to_head', 'house_owner_gender',
'years_to_complete_primary']

# Encode categorical variables
X = pd.get_dummies(X)

X = X.loc[:, ~X.columns.duplicated()]

column_list = X.columns.tolist()
print(column_list)

['age', 'first_prenatal_visit_month', 'has_property',
'total_household_income', 'household_ran_out_of_food',
'hungry_but_did_not_eat', 'monthly_income', 'num_prenatal_visits',
'num_rooms', 'prenatal_consultations', 'property_owner_gender',
'work_days_last_month', 'worried_about_food',
'years_to_complete_primary', 'birth_last_3yrs_3',
'birth_last_3yrs_rural', 'born_district_type_rural',
'born_district_type_urban', 'born_in_district_no',
'born_in_district_yes', 'can_do_math_no', 'can_do_math_yes',
'can_read_no', 'can_read_yes', 'can_report_income_annually',
'can_report_income_monthly', 'can_report_income_recieve only in kind',
'can_write_no', 'can_write_yes', 'computer_location_  home',
'computer_location_education place', 'connected_to_sewerage_no, no
system', 'connected_to_sewerage_yes under ground drain',
'connected_to_sewerage_yes, to covered drain',
'connected_to_sewerage_yes, to open drain', 'cooking_water_source_
motor pump / tube well', 'cooking_water_source_bottled water',
'cooking_water_source_filtration plant', 'cooking_water_source_hand
pump', 'cooking_water_source_others(specify_____________)',
'cooking_water_source_piped water', 'cooking_water_source_pond/canal /
river / stream', 'disability_No', 'disability_Yes',
'dwelling_type_apartment/flat', 'dwelling_type_independent
house/compound', 'dwelling_type_other (specify)', 'dwelling_type_part
of a compound', 'dwelling_type_part of the large unit',
```

```
'employment_status_5', 'employment_status_9', 'employment_status_no
(no difficulty)', 'employment_status_yes (alot of difficulty)',
"employment_status_yes (can't do at all)", 'employment_status_yes
(some difficulty)', 'gender_female', 'gender_male',
'handwashing_water_source_ motor pump / tube well',
'handwashing_water_source_closed well', 'handwashing_water_source_hand
pump', 'handwashing_water_source_open well',
'handwashing_water_source_others (specify----------------------)',
'handwashing_water_source_piped water', 'has_computer_no ',
'has_computer_yes ', 'has_handwashing_place_no',
'has_handwashing_place_yes', 'has_internet_no ', 'has_internet_yes ',
'has_job_no not seeking work', 'has_job_not but seeking work',
'has_mobile_mobile phone', 'has_mobile_none of above',
'has_mobile_smart phone', 'has_mobile_phone_no ',
'has_mobile_phone_yes ', 'house_owner_gender_dont know',
'house_owner_gender_female', 'house_owner_gender_jointly',
'house_owner_gender_male', 'household_member_no',
'household_member_yes', 'income_used_for_hh_no',
'income_used_for_hh_no income reported', 'income_used_for_hh_yes',
'marital_status_currently married', 'marital_status_unmarried / never
married', 'migration_reason_accompany family',
'migration_reason_better economic opportunities',
'migration_reason_education', 'migration_reason_marriage',
'migration_reason_others', 'no_computer_reason_affordability',
'no_computer_reason_do not use it because (not useful, not
interested,cultural reasons', 'no_computer_reason_don’t know how to
use it', 'no_computer_reason_other specify',
'no_computer_reason_privacy/security concerns',
'no_computer_reason_use substitutes instead like mobile
phone/smartphone etc', 'no_mobile_reason_cost of mobile is too high',
'no_mobile_reason_do not need the mobile (not useful)',
'no_mobile_reason_don’t know how to use mobile', 'no_mobile_reason_not
allowed to use mobile', 'no_mobile_reason_other reason',
'no_mobile_reason_privacy or security concerns',
'no_mobile_reason_service is not available in the area',
'no_mobile_reason_using land line', 'pay_for_water_ yes',
'pay_for_water_no', 'prenatal_consultation_source_yes (some
difficulty)', 'reason_for_headship_family elder',
'reason_for_headship_is oldest male in the house',
'reason_for_headship_main decision maker', 'reason_for_headship_main
economic provider', 'reason_for_headship_main provider away for work',
'reason_for_headship_other (specify)....', 'region_rural',
'region_urban', 'relationship_to_head_ nephew/niece',
'relationship_to_head_brother/sister', 'relationship_to_head_grand
child', 'relationship_to_head_head', 'relationship_to_head_others
(specify).........', 'relationship_to_head_son/daughter',
'relationship_to_head_son/daughter in law',
'relationship_to_head_spouse', 'residence_status_absent',
'residence_status_present', 'sick_last_2wks_rural',
```

```
'sick_last_2wks_urban', 'shared_toilet_0', 'shared_toilet_no',
'shared_toilet_yes', 'sufficient_drinking_water_ yes',
'sufficient_drinking_water_no', 'toilet_type_flush connected  to open
drain', 'toilet_type_flush connected to pit', 'toilet_type_flush
connected to public sewerage', 'toilet_type_flush connected to septic
tank', 'toilet_type_no toilet', 'toilet_type_other
(specify_____)', 'used_computer_desktop',
'used_computer_laptop ', 'used_computer_no', 'used_computer_tablet ',
'used_mobile_mobile phone', 'used_mobile_none of above',
'used_mobile_smart phone', 'worked_last_month_no',
'worked_last_month_yes']
```

```
duplicates = X.columns[X.columns.duplicated()]
print(duplicates)
```

```
Index([], dtype='object')
```

```
X = X[X.columns.drop(list(X.filter(regex='-1')))]
```

```
from sklearn.preprocessing import MinMaxScaler
 # Assuming X is a pandas DataFrame
scaler = MinMaxScaler()
 # Convert the scaled array back to a DataFrame (optional)
X = pd.DataFrame(scaler.fit_transform(X), columns=X.columns)
 # Calculate correlations
correlations = X.corrwith(y)
```

```
C:\Users\PMLS\AppData\Local\Programs\Python\Python311\Lib\site-
packages\numpy\lib\_function_base_impl.py:2999: RuntimeWarning:
invalid value encountered in divide
  c /= stddev[:, None]
C:\Users\PMLS\AppData\Local\Programs\Python\Python311\Lib\site-
packages\numpy\lib\_function_base_impl.py:3000: RuntimeWarning:
invalid value encountered in divide
  c /= stddev[None, :]
```

```
column_list = X.columns.tolist()
print(column_list)
```

```
['age', 'first_prenatal_visit_month', 'has_property',
'total_household_income', 'household_ran_out_of_food',
'hungry_but_did_not_eat', 'monthly_income', 'num_prenatal_visits',
'num_rooms', 'prenatal_consultations', 'property_owner_gender',
'work_days_last_month', 'worried_about_food',
'years_to_complete_primary', 'birth_last_3yrs_3',
'birth_last_3yrs_rural', 'born_district_type_rural',
'born_district_type_urban', 'born_in_district_no',
'born_in_district_yes', 'can_do_math_no', 'can_do_math_yes',
'can_read_no', 'can_read_yes', 'can_report_income_annually',
```

```
'can_report_income_monthly', 'can_report_income_recieve only in kind',
'can_write_no', 'can_write_yes', 'computer_location_  home',
'computer_location_education place', 'connected_to_sewerage_no, no
system', 'connected_to_sewerage_yes under ground drain',
'connected_to_sewerage_yes, to covered drain',
'connected_to_sewerage_yes, to open drain', 'cooking_water_source_
motor pump / tube well', 'cooking_water_source_bottled water',
'cooking_water_source_filtration plant', 'cooking_water_source_hand
pump', 'cooking_water_source_others(specify_____________)',
'cooking_water_source_piped water', 'cooking_water_source_pond/canal /
river / stream', 'disability_No', 'disability_Yes',
'dwelling_type_apartment/flat', 'dwelling_type_independent
house/compound', 'dwelling_type_other (specify)', 'dwelling_type_part
of a compound', 'dwelling_type_part of the large unit',
'employment_status_5', 'employment_status_9', 'employment_status_no
(no difficulty)', 'employment_status_yes (alot of difficulty)',
"employment_status_yes (can't do at all)", 'employment_status_yes
(some difficulty)', 'gender_female', 'gender_male',
'handwashing_water_source_ motor pump / tube well',
'handwashing_water_source_closed well', 'handwashing_water_source_hand
pump', 'handwashing_water_source_open well',
'handwashing_water_source_others (specify----------------------)',
'handwashing_water_source_piped water', 'has_computer_no ',
'has_computer_yes ', 'has_handwashing_place_no',
'has_handwashing_place_yes', 'has_internet_no ', 'has_internet_yes ',
'has_job_no not seeking work', 'has_job_not but seeking work',
'has_mobile_mobile phone', 'has_mobile_none of above',
'has_mobile_smart phone', 'has_mobile_phone_no ',
'has_mobile_phone_yes ', 'house_owner_gender_dont know',
'house_owner_gender_female', 'house_owner_gender_jointly',
'house_owner_gender_male', 'household_member_no',
'household_member_yes', 'income_used_for_hh_no',
'income_used_for_hh_no income reported', 'income_used_for_hh_yes',
'marital_status_currently married', 'marital_status_unmarried / never
married', 'migration_reason_accompany family',
'migration_reason_better economic opportunities',
'migration_reason_education', 'migration_reason_marriage',
'migration_reason_others', 'no_computer_reason_affordability',
'no_computer_reason_do not use it because (not useful, not
interested,cultural reasons', 'no_computer_reason_don’t know how to
use it', 'no_computer_reason_other specify',
'no_computer_reason_privacy/security concerns',
'no_computer_reason_use substitutes instead like mobile
phone/smartphone etc', 'no_mobile_reason_cost of mobile is too high',
'no_mobile_reason_do not need the mobile (not useful)',
'no_mobile_reason_don’t know how to use mobile', 'no_mobile_reason_not
allowed to use mobile', 'no_mobile_reason_other reason',
'no_mobile_reason_privacy or security concerns',
'no_mobile_reason_service is not available in the area',
```

```
'no_mobile_reason_using land line', 'pay_for_water_ yes',
'pay_for_water_no', 'prenatal_consultation_source_yes (some
difficulty)', 'reason_for_headship_family elder',
'reason_for_headship_is oldest male in the house',
'reason_for_headship_main decision maker', 'reason_for_headship_main
economic provider', 'reason_for_headship_main provider away for work',
'reason_for_headship_other (specify)....', 'region_rural',
'region_urban', 'relationship_to_head_ nephew/niece',
'relationship_to_head_brother/sister', 'relationship_to_head_grand
child', 'relationship_to_head_head', 'relationship_to_head_others
(specify).........', 'relationship_to_head_son/daughter',
'relationship_to_head_son/daughter in law',
'relationship_to_head_spouse', 'residence_status_absent',
'residence_status_present', 'sick_last_2wks_rural',
'sick_last_2wks_urban', 'shared_toilet_0', 'shared_toilet_no',
'shared_toilet_yes', 'sufficient_drinking_water_ yes',
'sufficient_drinking_water_no', 'toilet_type_flush connected  to open
drain', 'toilet_type_flush connected to pit', 'toilet_type_flush
connected to public sewerage', 'toilet_type_flush connected to septic
tank', 'toilet_type_no toilet', 'toilet_type_other
(specify_____)', 'used_computer_desktop',
'used_computer_laptop ', 'used_computer_no', 'used_computer_tablet ',
'used_mobile_mobile phone', 'used_mobile_none of above',
'used_mobile_smart phone', 'worked_last_month_no',
'worked_last_month_yes']
```

```
 # Convert correlations to a DataFrame for better printing
corr_df =
pd.DataFrame(correlations.abs().sort_values(ascending=False),
columns=['Correlation'])
```

```
 # Print all correlations
with pd.option_context('display.max_rows', None,
'display.max_columns', None):
 print(corr_df)
```

```
                                                  Correlation
age                                                  0.446492
can_read_no                                          0.400005
worked_last_month_yes                                0.328464
can_do_math_no                                       0.288861
income_used_for_hh_yes                               0.285413
can_report_income_monthly                            0.282258
monthly_income                                       0.266787
has_mobile_none of above                             0.265518
employment_status_yes (can't do at all)              0.262653
total_household_income                               0.254241
has_mobile_mobile phone                              0.186237
employment_status_5                                  0.155814
income_used_for_hh_no income reported                0.144126
```

has_mobile_smart phone	0.142860
has_job_not but seeking work	0.133262
household_member_yes	0.120627
household_member_no	0.120627
worked_last_month_no	0.111077
can_do_math_yes	0.094512
has_handwashing_place_no	0.090695
has_handwashing_place_yes	0.090695
disability_Yes	0.083309
disability_No	0.083309
property_owner_gender	0.082542
has_job_no not seeking work	0.080911
used_computer_no	0.076210
employment_status_yes (alot of difficulty)	0.075736
used_mobile_none of above	0.071319
no_mobile_reason_do not need the mobile (not us...	0.068624
reason_for_headship_is oldest male in the house	0.066429
dwelling_type_part of the large unit	0.062268
reason_for_headship_main economic provider	0.061157
house_owner_gender_jointly	0.058803
employment_status_9	0.057805
marital_status_currently married	0.057805
marital_status_unmarried / never married	0.057805
shared_toilet_yes	0.056955
no_computer_reason_use substitutes instead like...	0.055518
shared_toilet_no	0.055283
relationship_to_head_grand child	0.050273
dwelling_type_independent house/compound	0.050066
toilet_type_flush connected to open drain	0.049789
house_owner_gender_male	0.048786
handwashing_water_source_ motor pump / tube well	0.048769
no_mobile_reason_other reason	0.048301
cooking_water_source_others(specify_____)	0.048182
handwashing_water_source_others (specify-------...	0.047186
employment_status_yes (some difficulty)	0.047186
can_report_income_annually	0.047186
relationship_to_head_spouse	0.047186
reason_for_headship_family elder	0.046535
no_mobile_reason_cost of mobile is too high	0.046465
cooking_water_source_hand pump	0.045807
sick_last_2wks_rural	0.045806
no_mobile_reason_not allowed to use mobile	0.045714
has_property	0.044876
work_days_last_month	0.044209
handwashing_water_source_hand pump	0.043852
relationship_to_head_brother/sister	0.043613
no_mobile_reason_don’t know how to use mobile	0.043502
no_computer_reason_affordability	0.042850
birth_last_3yrs_3	0.040860

has_internet_yes	0.039257
has_internet_no	0.039257
has_mobile_phone_no	0.037487
has_mobile_phone_yes	0.037487
reason_for_headship_main decision maker	0.036778
toilet_type_other (specify_____)	0.033358
income_used_for_hh_no	0.033358
birth_last_3yrs_rural	0.033358
prenatal_consultation_source_yes (some difficulty)	0.033358
used_mobile_mobile phone	0.033160
can_write_yes	0.032236
toilet_type_flush connected to septic tank	0.031609
cooking_water_source_filtration plant	0.031438
cooking_water_source_ motor pump / tube well	0.030081
born_in_district_yes	0.029351
toilet_type_flush connected to public sewerage	0.028339
has_computer_yes	0.028048
has_computer_no	0.028048
relationship_to_head_ nephew/niece	0.027388
used_computer_desktop	0.026747
can_read_yes	0.026654
used_computer_tablet	0.025353
used_mobile_smart phone	0.024829
house_owner_gender_dont know	0.024746
relationship_to_head_son/daughter	0.023905
employment_status_no (no difficulty)	0.023585
migration_reason_marriage	0.023585
relationship_to_head_head	0.023585
relationship_to_head_son/daughter in law	0.023585
migration_reason_education	0.023585
computer_location_education place	0.022964
reason_for_headship_main provider away for work	0.021433
no_computer_reason_privacy/security concerns	0.019742
dwelling_type_other (specify)	0.018741
can_write_no	0.018458
pay_for_water_ yes	0.017686
pay_for_water_no	0.017686
connected_to_sewerage_yes under ground drain	0.016742
region_rural	0.016694
region_urban	0.016694
connected_to_sewerage_yes, to covered drain	0.015918
no_computer_reason_don’t know how to use it	0.015027
toilet_type_flush connected to pit	0.014631
computer_location_ home	0.012391
born_district_type_rural	0.012298
connected_to_sewerage_no, no system	0.012029
handwashing_water_source_piped water	0.011857
dwelling_type_part of a compound	0.011305
residence_status_absent	0.010870

```
residence_status_present                                0.010870
migration_reason_better economic opportunities          0.010344
shared_toilet_0                                         0.010344
handwashing_water_source_open well                      0.010344
no_mobile_reason_using land line                        0.010344
migration_reason_others                                 0.010344
worried_about_food                                      0.010210
dwelling_type_apartment/flat                            0.009769
no_computer_reason_other specify                        0.009418
used_computer_laptop                                    0.009375
no_mobile_reason_service is not available in th...      0.009364
can_report_income_recieve only in kind                  0.009364
handwashing_water_source_closed well                    0.009364
no_mobile_reason_privacy or security concerns           0.009364
household_ran_out_of_food                               0.009029
num_rooms                                               0.008844
cooking_water_source_pond/canal / river / stream        0.007961
sick_last_2wks_urban                                    0.006437
born_in_district_no                                     0.006415
born_district_type_urban                                0.005153
house_owner_gender_female                               0.004899
migration_reason_accompany family                       0.004008
reason_for_headship_other (specify)....                 0.003725
cooking_water_source_bottled water                      0.003356
sufficient_drinking_water_ yes                          0.002972
sufficient_drinking_water_no                            0.002972
hungry_but_did_not_eat                                  0.002877
connected_to_sewerage_yes, to open drain                0.002829
toilet_type_no toilet                                   0.002662
relationship_to_head_others (specify).........          0.002366
cooking_water_source_piped water                        0.001878
no_computer_reason_do not use it because (not u...      0.001258
gender_female                                           0.000362
gender_male                                             0.000362
first_prenatal_visit_month                                   NaN
num_prenatal_visits                                          NaN
prenatal_consultations                                       NaN
years_to_complete_primary                                    NaN
```

```
# Filter for correlations with absolute value greater than 0.01
selected_features = correlations[abs(correlations) >= 0.01].index
print(len(X.columns))
print(len(selected_features))
```

```
149
118
```

```
# Filter for correlations with absolute value greater than 0.05
# Keep only the selected columns in X
```

```
 X = X[selected_features]
 print(len(X.columns))
```

```
118
```

```
# handling the missing values
missing_counts = X.isnull().sum()
thresh = 1000
cols_to_remove = missing_counts[missing_counts >
thresh].index.tolist()
print(cols_to_remove)
```

```
['total_household_income', 'monthly_income', 'property_owner_gender',
'work_days_last_month']
```

```
X = X.drop(columns=cols_to_remove)

#encoding the catagorical variables
y = pd.get_dummies(y)



from sklearn.model_selection import train_test_split
#splitting the data
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

Making Random Forest Model

```
import numpy as np

y_train = np.argmax(y_train, axis=1)
y_test = np.argmax(y_test, axis=1)

import numpy as np
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report


# Initialize the Random Forest Classifier
rf_model = RandomForestClassifier(n_estimators=100,  # number of trees
                                  random_state=42,
                                  n_jobs=-1)         # use all
processors

# Train the model
rf_model.fit(X_train, y_train)
```

```python
# Make predictions on the test set
y_pred = rf_model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

# Print a detailed classification report
print(classification_report(y_test, y_pred))

# Optionally, display the confusion matrix
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", cm)
```

```
Accuracy: 0.9024390243902439
              precision    recall  f1-score   support

           0       0.91      0.76      0.83       252
           1       0.90      0.97      0.93       568

    accuracy                           0.90       820
   macro avg       0.90      0.86      0.88       820
weighted avg       0.90      0.90      0.90       820

Confusion Matrix:
 [[191  61]
 [ 19 549]]
```

```
y_train shape: (3280,)
y_test shape: (820,)
```

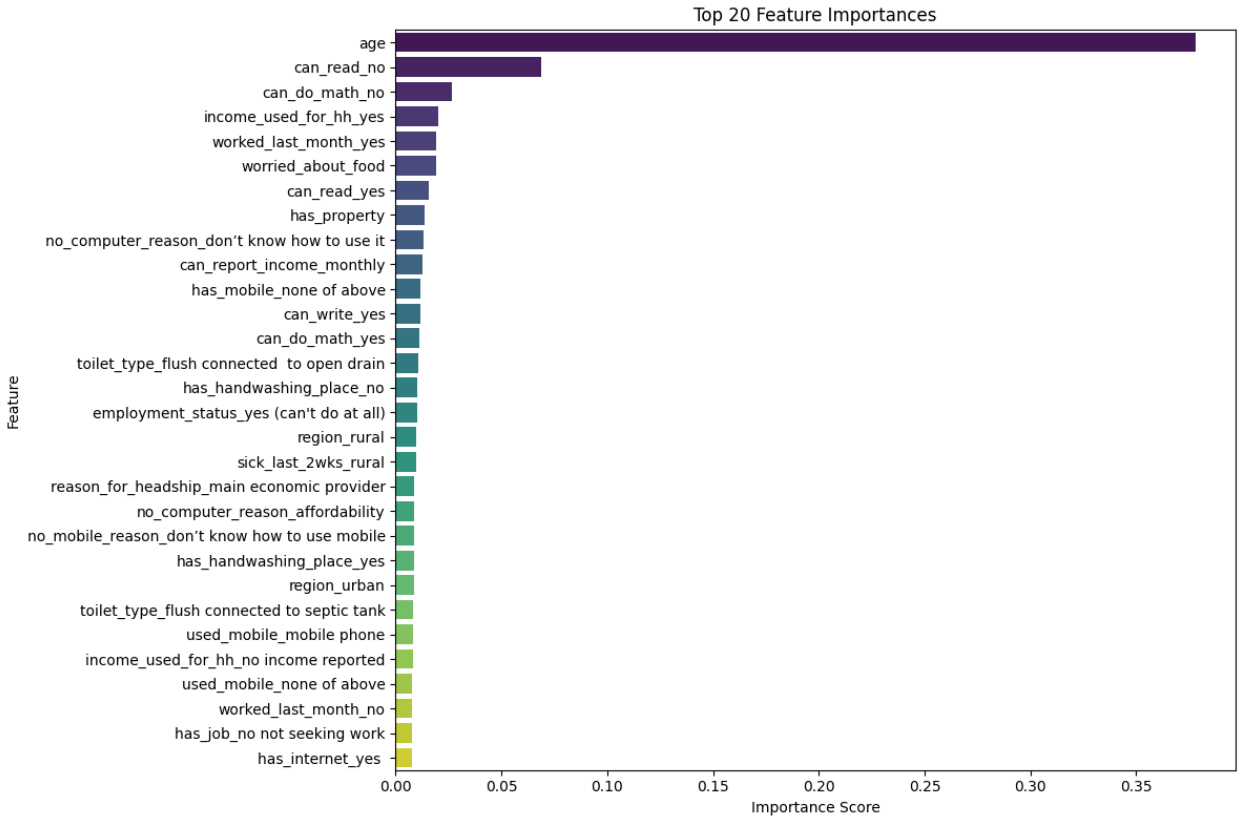
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Create a DataFrame with feature names and their corresponding
importance scores
feat_importances = pd.DataFrame({
    'feature': X_train.columns,
    'importance': rf_model.feature_importances_
})

# Sort the DataFrame in descending order of importance
feat_importances = feat_importances.sort_values(by='importance',
ascending=False)

# Get the top 20 features
top30 = feat_importances.head(30)
```

```python
# Print the top 20 features
print(top30)

# Optional: visualize the top 20 features using a horizontal bar plot
plt.figure(figsize=(12,8))
sns.barplot(x='importance', y='feature', data=top30,
palette="viridis")
plt.title("Top 20 Feature Importances")
plt.xlabel("Importance Score")
plt.ylabel("Feature")
plt.tight_layout()
plt.show()
```

	feature	importance
0	age	0.378083
9	can_read_no	0.068856
7	can_do_math_no	0.026855
61	income_used_for_hh_yes	0.020322
113	worked_last_month_yes	0.019332
2	worried_about_food	0.019100
10	can_read_yes	0.015492
1	has_property	0.013506
69	no_computer_reason_don’t know how to use it	0.013389
12	can_report_income_monthly	0.012715
50	has_mobile_none of above	0.011881
14	can_write_yes	0.011851
8	can_do_math_yes	0.011348
101	toilet_type_flush connected to open drain	0.010660
43	has_handwashing_place_no	0.010258
34	employment_status_yes (can't do at all)	0.010200
86	region_rural	0.009608
97	sick_last_2wks_rural	0.009555
84	reason_for_headship_main economic provider	0.008833
68	no_computer_reason_affordability	0.008751
74	no_mobile_reason_don’t know how to use mobile	0.008703
44	has_handwashing_place_yes	0.008612
87	region_urban	0.008583
104	toilet_type_flush connected to septic tank	0.008360
109	used_mobile_mobile phone	0.008267
60	income_used_for_hh_no income reported	0.008236
110	used_mobile_none of above	0.007872
112	worked_last_month_no	0.007758
47	has_job_no not seeking work	0.007751
46	has_internet_yes	0.007679

```
C:\Users\PMLS\AppData\Local\Temp\ipykernel_24712\3542814801.py:22:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
```

```
removed in v0.14.0. Assign the `y` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x='importance', y='feature', data=top30,
palette="viridis")
```

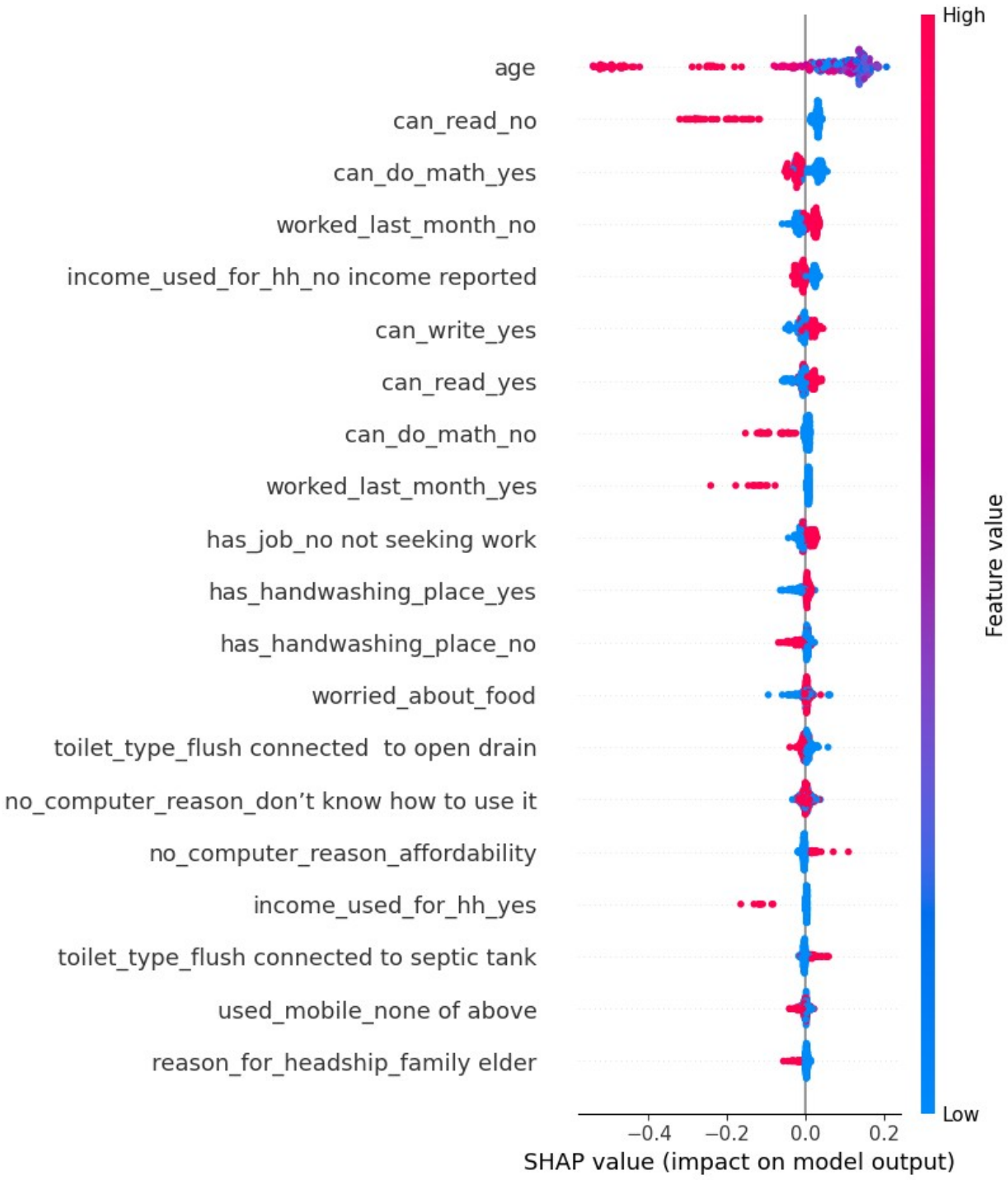



```
import shap

C:\Users\PMLS\AppData\Local\Programs\Python\Python311\Lib\site-
packages\tqdm\auto.py:21: TqdmWarning: IProgress not found. Please
update jupyter and ipywidgets. See
https://ipywidgets.readthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm

# explainer = shap.Explainer(model, X_train)
# shap_values = explainer.shap_values(X_test)

PermutationExplainer explainer: 821it [05:56,  2.25it/s]


# test_data = X_test

import shap
import matplotlib.pyplot as plt
import pandas as pd
```

```python
# ------------------------------
# 1. Create a subset of your test data for visualization
# ------------------------------
# Use the first 300 samples for faster plotting.
X_test_subset = X_test.iloc[:300]

# ------------------------------
# 2. Initialize the Explainer using the universal SHAP Explainer for your Random Forest model
# ------------------------------
explainer = shap.Explainer(rf_model, X_train)

# ------------------------------
# 3. Compute SHAP values on the entire test set, disabling the additivity check.
# ------------------------------
shap_values = explainer(X_test, check_additivity=False)
print("Full SHAP values shape:", shap_values.values.shape)  # Expected: (820, n_features, n_classes)

# ------------------------------
# 4. Extract SHAP values for the positive class (index 1)
# ------------------------------
shap_values_class1 = shap_values.values[:, :, 1]
print("SHAP values for class 1 shape:", shap_values_class1.shape)  # Expected: (820, n_features)

# ------------------------------
# 5. Subset the SHAP values to match the subset of test data
# ------------------------------
shap_values_subset = shap_values_class1[:300, :]
print("Subset SHAP values shape:", shap_values_subset.shape)  # Expected: (300, n_features)

# ------------------------------
# 6. Create the SHAP summary plot for feature importance using the subset
# ------------------------------
plt.figure(figsize=(10, 8))
shap.summary_plot(
    shap_values_subset,         # SHAP values for the positive class, subset to 300 samples
    X_test_subset,              # Subset of test data for visualization
    feature_names=X_train.columns  # Feature names from your training data
)
plt.tight_layout()
```

```
plt.savefig('shap_summary_plot.png', dpi=300)
plt.show()
```

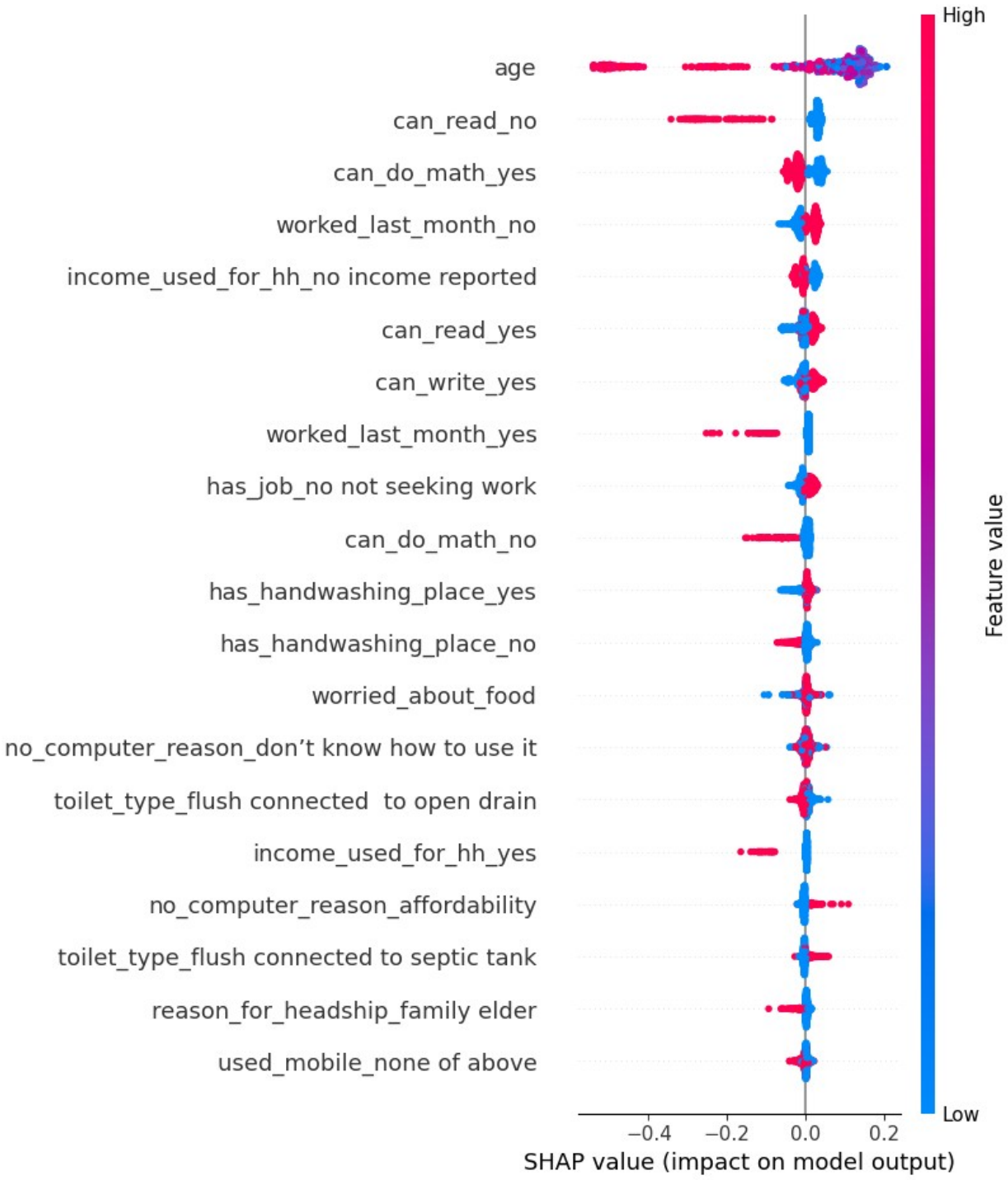
```
100%|==================| 1637/1640 [03:28<00:00]

Full SHAP values shape: (820, 114, 2)
SHAP values for class 1 shape: (820, 114)
Subset SHAP values shape: (300, 114)
```




```
<Figure size 640x480 with 0 Axes>
```

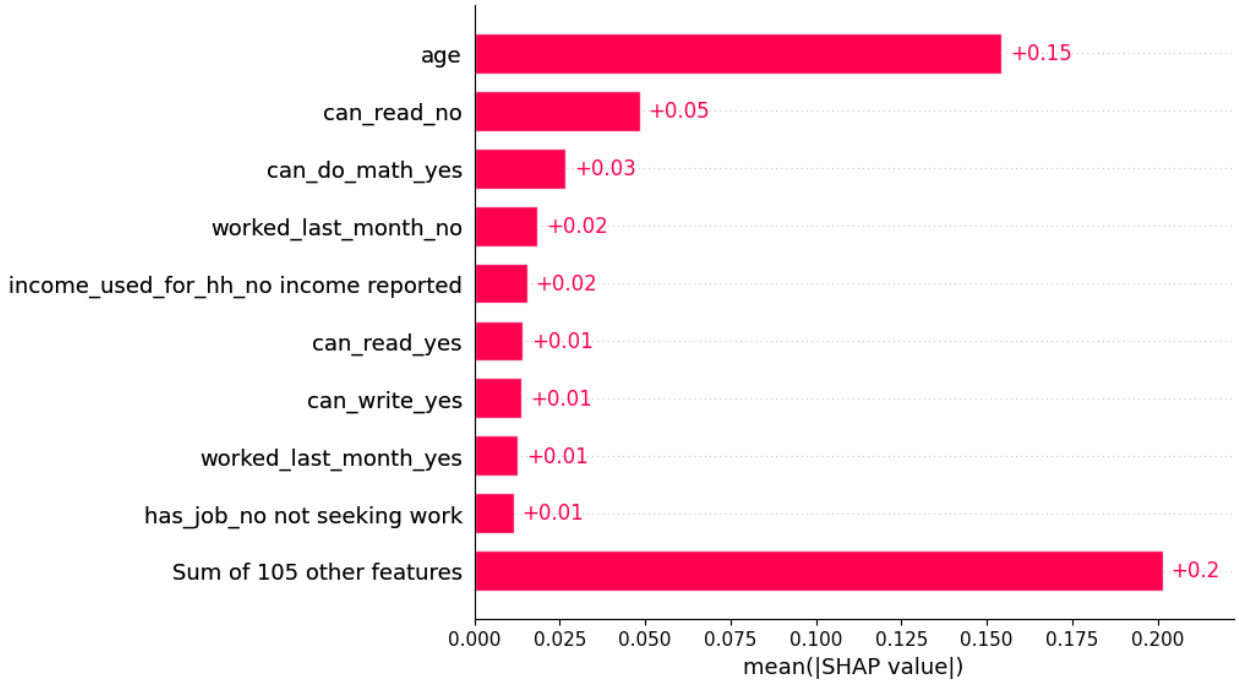
```
import shap
import matplotlib.pyplot as plt
import pandas as pd

# ------------------------------
```

```
# 1. Use the entire test set for visualization
# ------------------------------
X_test_subset = X_test  # Using the entire test set

# ------------------------------
# 2. Initialize the Explainer for your Random Forest model
# ------------------------------
explainer = shap.Explainer(rf_model, X_train)

# ------------------------------
# 3. Compute SHAP values on the entire test set, disabling the
additivity check.
# ------------------------------
shap_values = explainer(X_test, check_additivity=False)
print("Full SHAP values shape:", shap_values.values.shape)
# Expected shape: (n_samples, n_features, n_classes) e.g. (820, 114,
2)

# ------------------------------
# 4. Extract SHAP values for the positive class (index 1)
# ------------------------------
shap_values_class1 = shap_values.values[:, :, 1]
print("SHAP values for class 1 shape:", shap_values_class1.shape)
# Expected shape: (n_samples, n_features)

# ------------------------------
# 5. Use all SHAP values (matching the entire test set)
# ------------------------------
shap_values_subset = shap_values_class1  # using entire test set

# ------------------------------
# 6. Create the SHAP summary plot for feature importance using the
full test set
# ------------------------------
plt.figure(figsize=(10, 8))
shap.summary_plot(
    shap_values_subset,         # SHAP values for the positive class
for all samples
    X_test_subset,              # The full test set
    feature_names=X_train.columns  # Feature names from your training
data
)
plt.tight_layout()
plt.savefig('shap_summary_plot_full.png', dpi=300)
plt.show()

100%|==================| 1638/1640 [03:44<00:00]

Full SHAP values shape: (820, 114, 2)
SHAP values for class 1 shape: (820, 114)
```

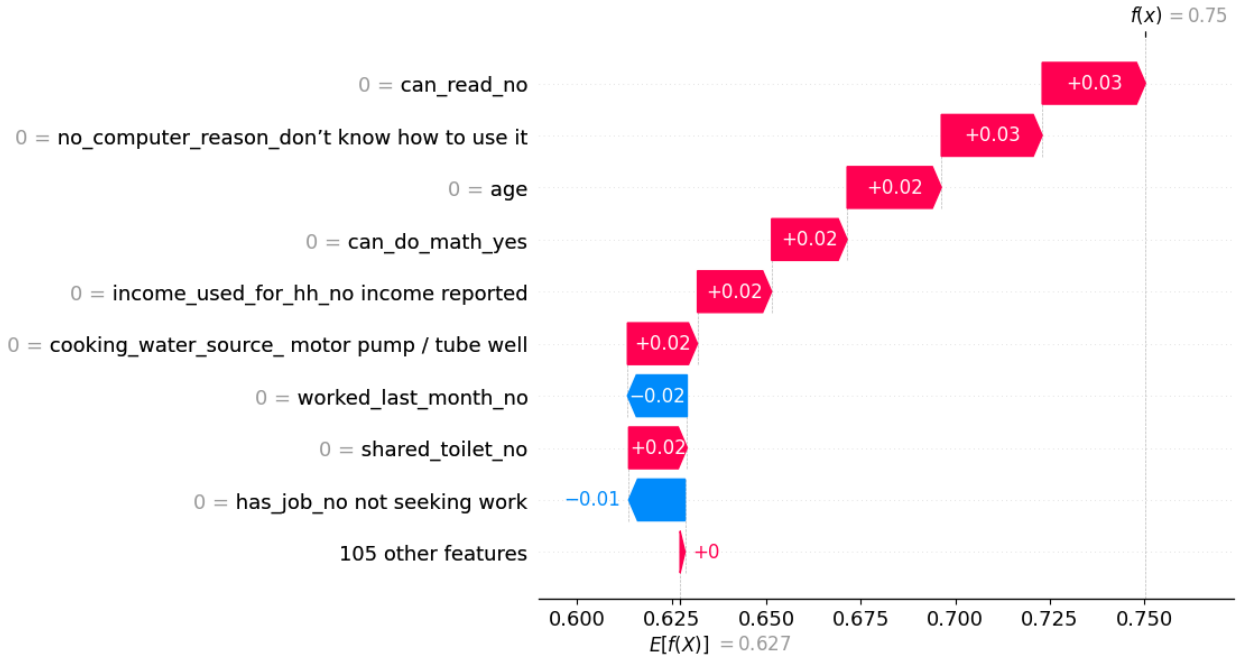



```
<Figure size 640x480 with 0 Axes>

shap_values_exp = shap.Explanation(
    values=shap_values_subset,
    base_values=explainer.expected_value[1],
    data=X_test_subset,
```

```
    feature_names=X_train.columns.to_list()
)

plt.figure(figsize=(8, 6))  # Adjust figure size if needed
shap.plots.bar(shap_values_exp)

plt.savefig('shap_bar_plot.png', dpi=150, bbox_inches='tight')  # Save
the plot
plt.show()
```

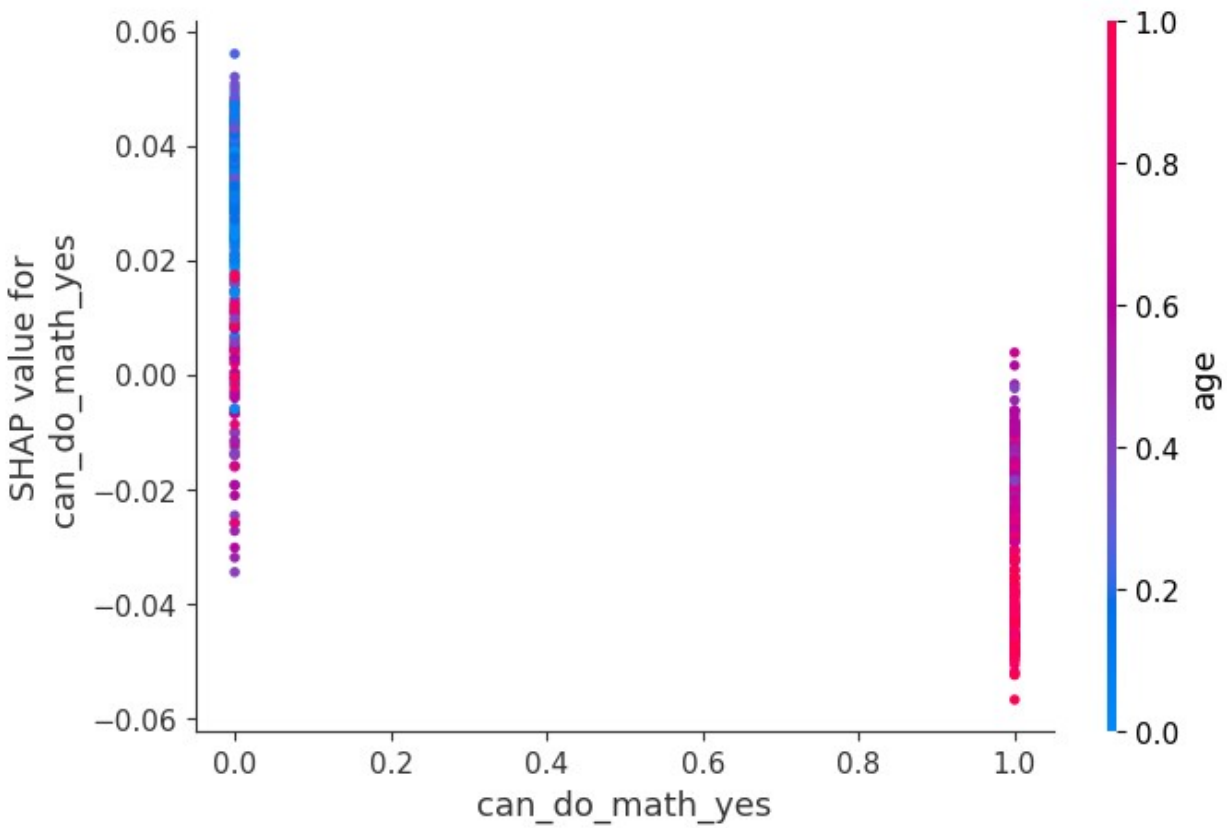



```
<Figure size 640x480 with 0 Axes>

shap.waterfall_plot(
    shap.Explanation(
        values=shap_values_subset[0],  # SHAP values for the first
instance
        base_values=explainer.expected_value[1],  # Expected model
output for class 1
        data=X_test_subset.iloc[0],  # Feature values for the first
instance
        feature_names=X_train.columns.to_list()
    )
)
plt.savefig('shap_waterfall_plot.png', dpi=150)
plt.show()
```



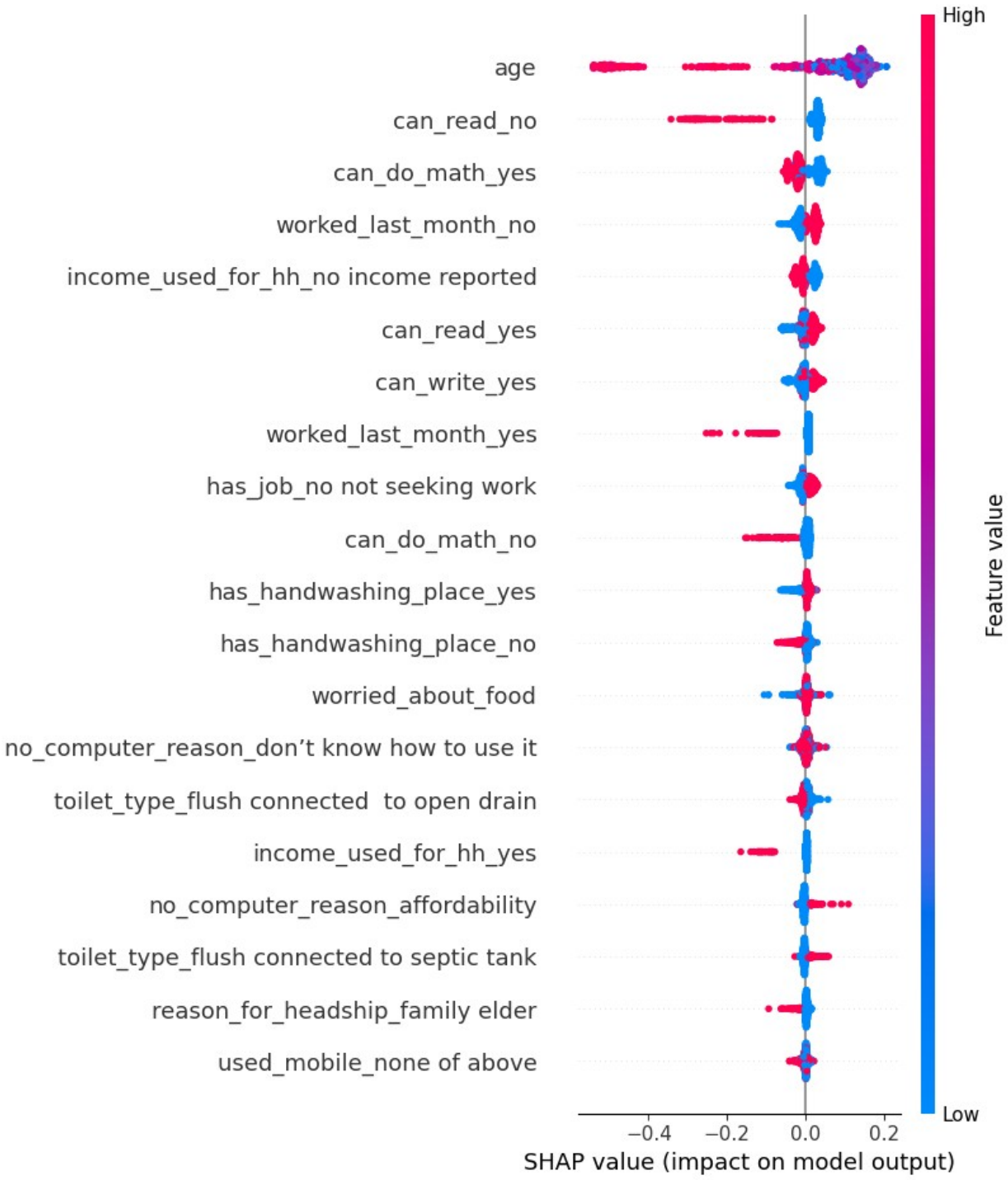

```
<Figure size 640x480 with 0 Axes>
```

```
feature_to_analyze = "can_do_math_yes"  # Replace with an actual
feature name
shap.dependence_plot(
    feature_to_analyze,
    shap_values_subset,
    X_test_subset
)
plt.savefig(f'shap_dependence_{feature_to_analyze}.png', dpi=150)
plt.show()
```




```
<Figure size 640x480 with 0 Axes>
```

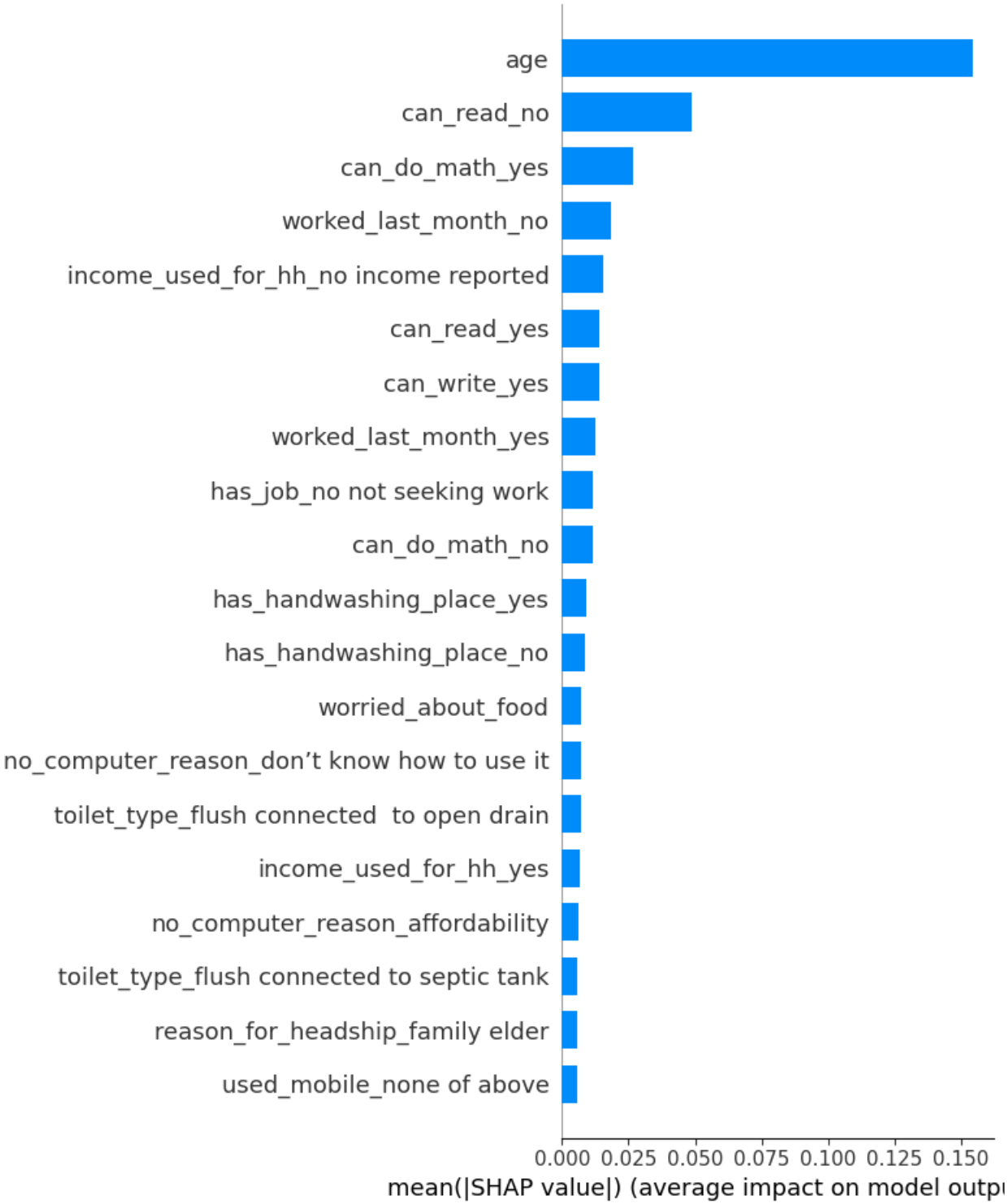
```
plt.figure(figsize=(8, 6))
shap.summary_plot(shap_values_subset, X_test_subset)
plt.savefig('shap_beeswarm_plot.png', dpi=150)
plt.show()
```




```
<Figure size 640x480 with 0 Axes>
```

```
plt.figure(figsize=(8, 6))
shap.summary_plot(
    shap_values_subset,
    X_test_subset,
```

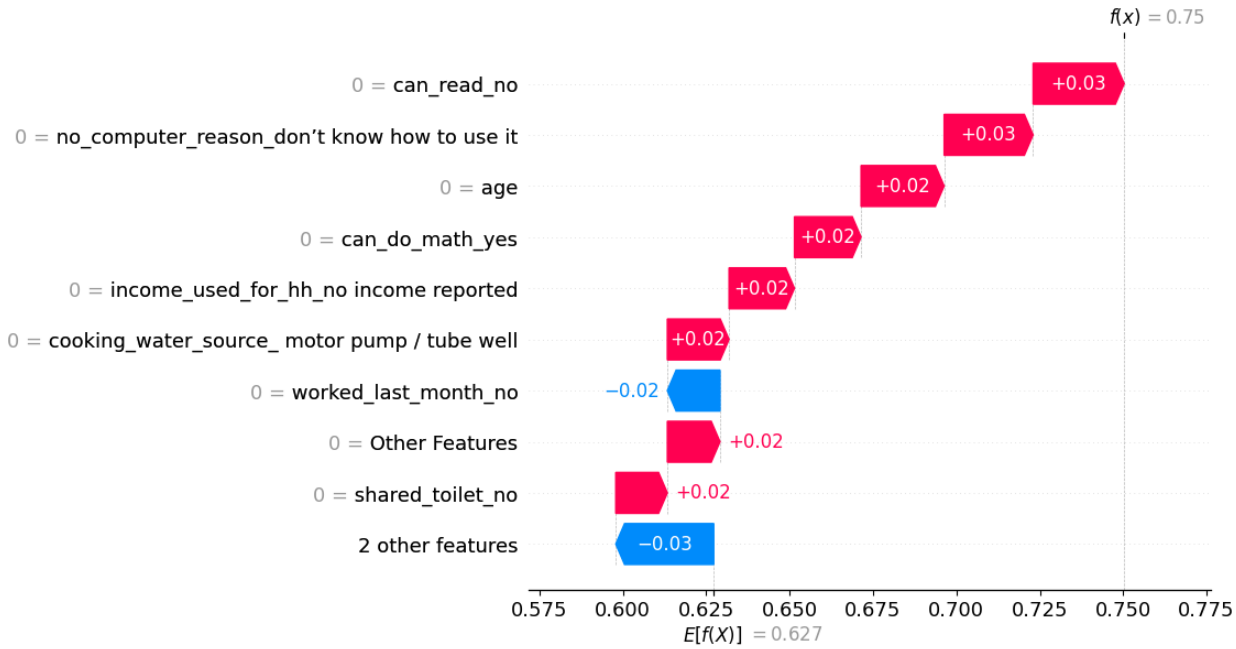
```
    feature_names=X_train.columns,
    plot_type="bar"
)
plt.tight_layout()
plt.savefig('shap_feature_importance_bar.png', dpi=150)
plt.show()
```




```
<Figure size 640x480 with 0 Axes>
```

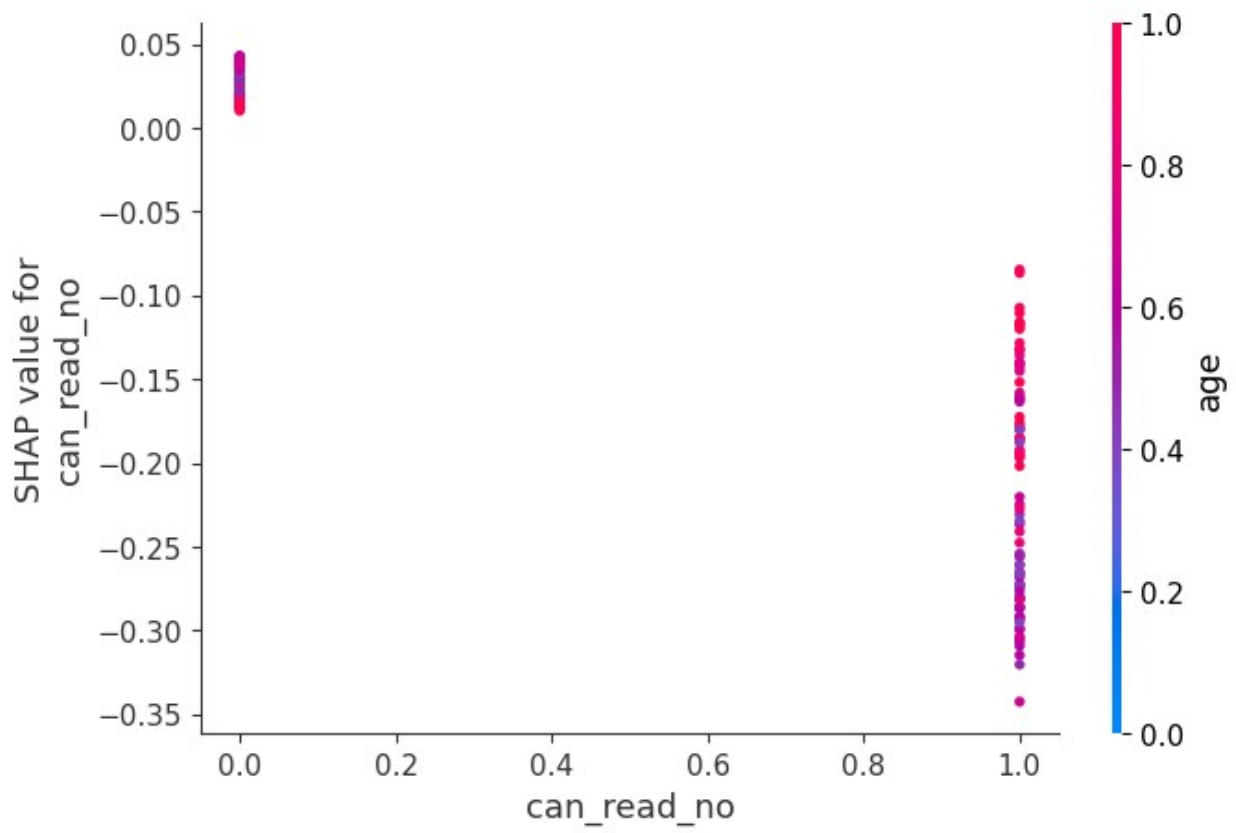
```
import shap
import matplotlib.pyplot as plt
import numpy as np
```

```python
# Number of top features to show in the waterfall plot
top_n = 10

# Extract SHAP values for the first sample
shap_vals = shap_values_subset[0]
feature_names = X_train.columns.to_list()
base_value = explainer.expected_value[1]

# Sort SHAP values by absolute importance
sorted_indices = np.argsort(np.abs(shap_vals))[::-1]  # Descending
order
shap_vals_sorted = shap_vals[sorted_indices]
feature_names_sorted = [feature_names[i] for i in sorted_indices]
feature_values_sorted = X_test_subset.iloc[0, sorted_indices]

# Keep only top N features, sum the rest into "Other Features"
if len(shap_vals_sorted) > top_n:
    other_features_sum = np.sum(shap_vals_sorted[top_n:])
    shap_vals_trimmed = np.append(shap_vals_sorted[:top_n],
other_features_sum)
    feature_names_trimmed = feature_names_sorted[:top_n] + ["Other
Features"]
    feature_values_trimmed = np.append(feature_values_sorted[:top_n],
0)  # No specific value for "Other Features"
else:
    shap_vals_trimmed = shap_vals_sorted
    feature_names_trimmed = feature_names_sorted
    feature_values_trimmed = feature_values_sorted

# Create the SHAP Waterfall Plot
shap.waterfall_plot(
    shap.Explanation(
        values=shap_vals_trimmed,
        base_values=base_value,
        data=feature_values_trimmed,
        feature_names=feature_names_trimmed
    )
)

# Save the figure
plt.savefig('shap_waterfall_plot_modified.png', dpi=150)
plt.show()
```

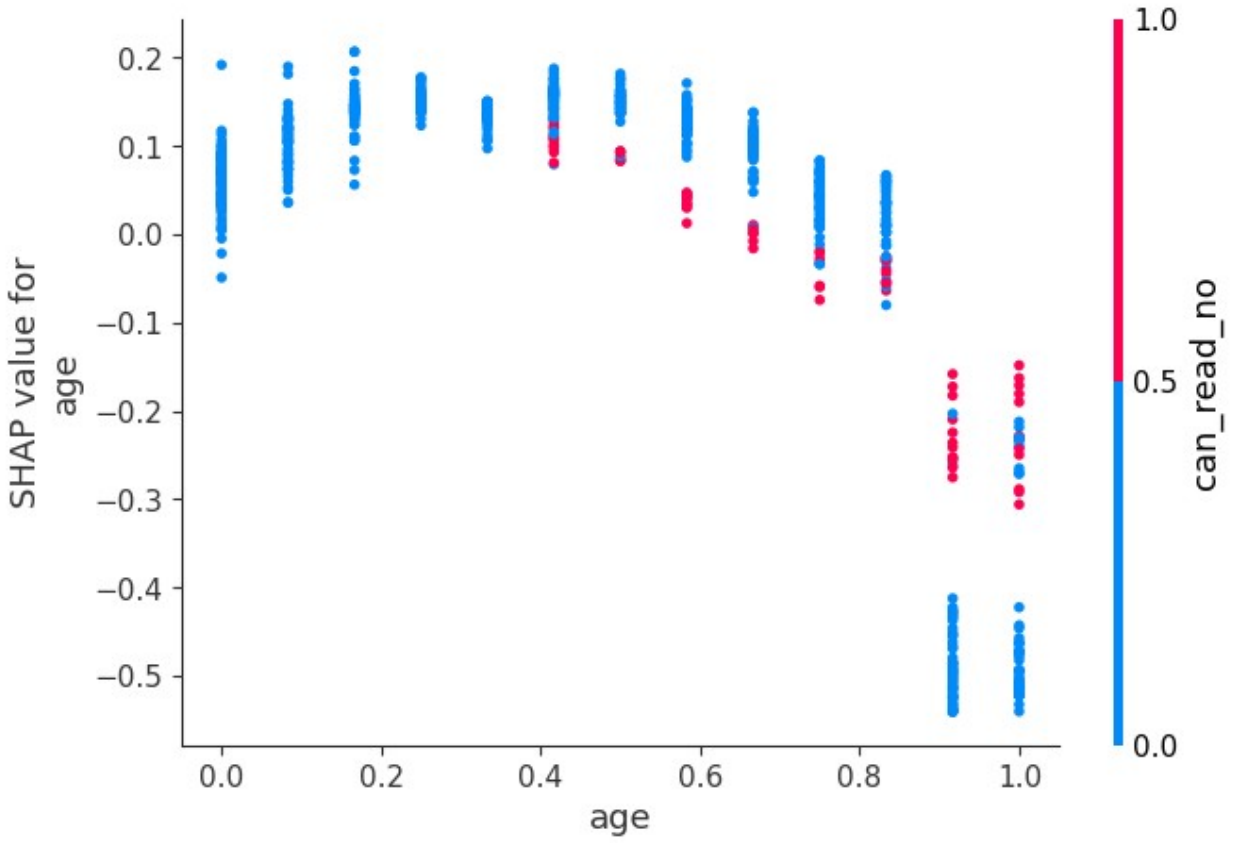



```
<Figure size 640x480 with 0 Axes>
```

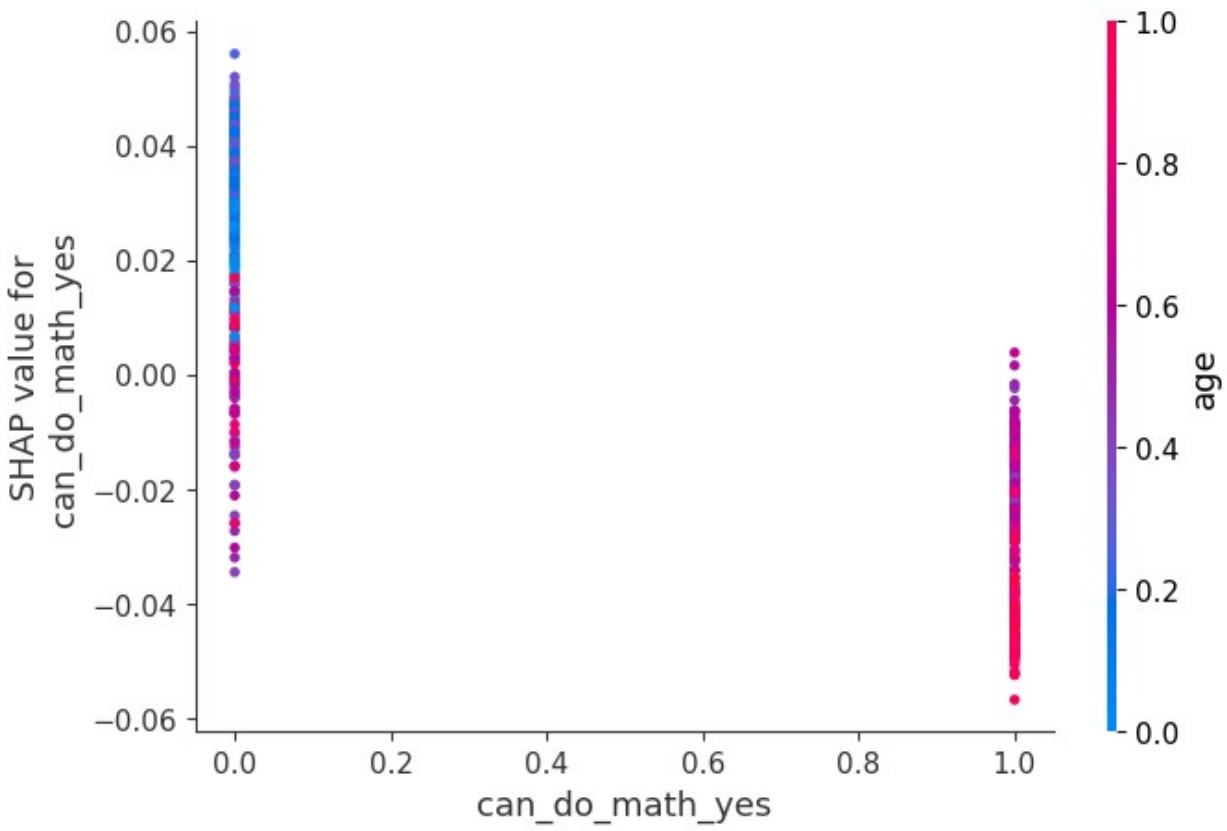
```
shap.dependence_plot("can_read_no", shap_values_subset, X_test_subset)
```



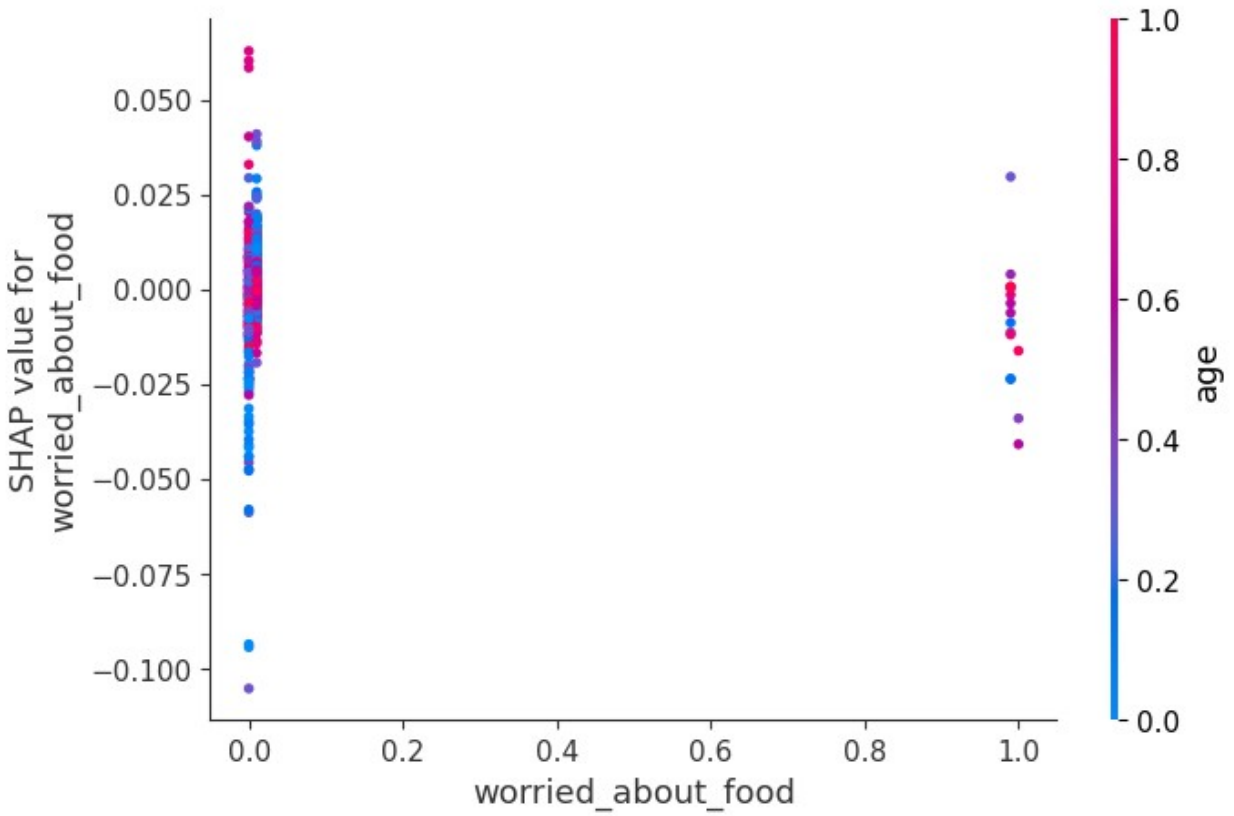
```
shap.dependence_plot("age", shap_values_subset, X_test_subset)
```




```
shap.dependence_plot("can_do_math_yes", shap_values_subset,
X_test_subset)
```




```
shap.dependence_plot("worried_about_food", shap_values_subset,
X_test_subset)
```


SHAP value for
worried_about_food
0.050
0.025
0.000
−0.025
−0.050
−0.075
−0.100
0.0
0.2
0.4
0.6
0.8
1.0
worried_about_food
1.0
0.8
0.6
0.4
0.2
0.0
age