practice

December 27, 2023

[2]:
```python
import pandas as pd

tomato = pd.read_csv('/Users/thutranghoa/Code/Data_analysis/Data/
 ↪1666946960_727__tomato-yields.csv')
tomato
```

[2]:
```
         Entity Code  Year  \
0        Africa  NaN  1961
1        Africa  NaN  1962
2        Africa  NaN  1963
3        Africa  NaN  1964
4        Africa  NaN  1965
…           …   …     …
11277  Zimbabwe  ZWE  2016
11278  Zimbabwe  ZWE  2017
11279  Zimbabwe  ZWE  2018
11280  Zimbabwe  ZWE  2019
11281  Zimbabwe  ZWE  2020

       Tomatoes | 00000388 || Yield | 005419 || tonnes per hectare
0                                              12.320172
1                                              12.976988
2                                              12.867894
3                                              13.189582
4                                              13.492712
…                                                  …
11277                                           7.237900
11278                                           7.219100
11279                                           7.225900
11280                                           7.226900
11281                                           7.224700

[11282 rows x 4 columns]
```

[9]:
```python
tomato.isna().sum()
```

[9]:
```
Entity                                                       0
Code                                                      2489
Year                                                         0
Tomatoes | 00000388 || Yield | 005419 || tonnes per hectare       0
dtype: int64
```

[17]:
```
tomato.describe()
```

[17]:

	Year	Yields
count	11282.000000	11282.000000
mean	1992.067364	36.514492
std	17.225589	61.719855
min	1961.000000	0.467600
25%	1977.000000	10.140050
50%	1993.000000	18.015349
75%	2007.000000	35.298426
max	2020.000000	523.927795

[10]:
```
tomato = tomato.rename(columns={"Tomatoes | 00000388 || Yield | 005419 ||␣
↪tonnes per hectare" : "Yields"})
```

[11]:
```
tomato.head(5)
```

[11]:

	Entity	Code	Year	Yields
0	Africa	NaN	1961	12.320172
1	Africa	NaN	1962	12.976988
2	Africa	NaN	1963	12.867894
3	Africa	NaN	1964	13.189582
4	Africa	NaN	1965	13.492712

[12]:
```
new_tomato = tomato.copy()
new_tomato = new_tomato.pivot(index = 'Entity',columns='Year', values='Yields')
new_tomato
```

[12]:

Year	1961	1962	1963	1964	1965 \
Entity					
Africa	12.320172	12.976988	12.867894	13.189582	13.492712
Africa (FAO)	12.336499	12.962899	12.888400	13.195300	13.452499
Albania	12.000000	12.000000	12.400000	12.799999	12.799999
Algeria	16.456999	17.500000	17.500000	13.644899	12.285299
Americas (FAO)	18.990599	21.422100	19.558899	20.495800	22.032900
…	…	…	…	…	…
World	16.434599	16.976000	16.734600	17.468300	18.143700
Yemen	NaN	NaN	NaN	NaN	NaN
Yugoslavia	12.035399	12.398399	11.910600	12.177400	10.801499
Zambia	10.000000	10.000000	10.000000	10.000000	10.000000
Zimbabwe	7.222200	7.157900	7.368400	7.000000	7.500000

```
Year                 1966       1967       1968       1969      1970  …  \
Entity                                                                 …
Africa          13.327377  12.466840  12.748196  13.281709  12.92659  …
Africa (FAO)    13.269099  12.445000  12.705600  13.228399  12.89090  …
Albania         12.500000  13.214299  12.857100  12.000000  12.33330  …
Algeria         11.177899   9.095799  10.047800  11.190700   9.44960  …
Americas (FAO)  21.345299  21.999800  24.566200  22.137699  23.05410  …
…                     …         …          …          …        …  …
World           18.300299  18.771700  19.288700  18.878700  19.32810  …
Yemen                 NaN        NaN        NaN        NaN       NaN  …
Yugoslavia      11.443299  10.841800  10.419399  10.405499   9.47530  …
Zambia          10.000000  10.000000  10.000000  11.250000  10.88240  …
Zimbabwe         7.272700   7.272700   7.272700   7.217400   7.21740  …

Year                 2011       2012       2013       2014       2015  \
Entity
Africa          19.040854  16.706995  15.755281  17.457846  17.360165
Africa (FAO)    18.850000  16.706999  15.755300  17.457800  17.360199
Albania         32.786900  31.538500  36.022301  37.184399  41.082298
Algeria         37.502098  36.995800  43.342400  47.055099  48.359299
Americas (FAO)  53.996197  56.599800  56.908497  60.573200  59.540798
…                     …         …          …          …        …
World           34.805698  33.974800  34.083599  35.511799  36.588100
Yemen           13.276700  14.154200  11.201500  11.313900  15.515400
Yugoslavia            NaN        NaN        NaN        NaN        NaN
Zambia           9.771999  10.000000   9.863000   9.753699   9.770700
Zimbabwe         7.151900   7.121200   7.121200   7.202100   7.218600

Year                 2016       2017       2018       2019       2020
Entity
Africa          14.999870  14.249650  13.247025  13.902744  14.087778
Africa (FAO)    14.999900  14.249599  13.247000  13.902699  14.087800
Albania         44.014198  44.370499  43.817497  44.975098  45.649399
Algeria         56.772900  53.646698  58.672398  59.124599  62.164700
Americas (FAO)  58.476498  57.541199  62.518597  65.266701  67.644997
…                     …         …          …          …        …
World           36.540199  36.509197  36.013500  36.608997  36.979797
Yemen           13.285000  13.340600  14.188900  13.443900  13.219299
Yugoslavia            NaN        NaN        NaN        NaN        NaN
Zambia           9.744699   9.762000   9.787300   9.785000   9.786400
Zimbabwe         7.237900   7.219100   7.225900   7.226900   7.224700

[220 rows x 60 columns]
```

0.0.1 Quoc gia co san luong lon nhat 2000

[55]:
```
'Quoc gia co san luong lon nhat 2000'
year_2000 = tomato.loc[tomato['Year'] == 2000]
df = year_2000.sort_values(by = ['Yields'], ascending=False)
print (df)
print ('Quốc gia có sản lượng cà chua lớn nhất năm 2000 là : ', df.
 ↪iloc[0]['Entity'])
```

```
                 Entity Code  Year      Yields
6762        Netherlands  NLD  2000  433.333282
2641            Denmark  DNK  2000  392.592590
10555    United Kingdom  GBR  2000  377.000000
9626             Sweden  SWE  2000  353.061188
7391             Norway  NOR  2000  328.032288
…                   …   …    …         …
7743   Papua New Guinea  PNG  2000    4.750000
10046              Togo  TGO  2000    4.046200
339              Angola  AGO  2000    3.714300
9986              Timor  TLS  2000    2.928600
8846            Somalia  SOM  2000    1.532400

[207 rows x 4 columns]
Quốc gia có sản lượng cà chua lớn nhất năm 2000 là :  Netherlands
```

Trong thoi 1961 - 2000, quoc gia nao co san luong lon nhat

[56]:
```
year_1961_2000 = tomato.loc[(tomato['Year'] >= 1961) & tomato['Year'] <=2000]
temp = year_1961_2000.groupby('Entity', as_index=False).sum()
temp = temp.drop(['Code', 'Year'], axis=1)
temp = temp.sort_values(by = ['Yields'], ascending=False)
print (temp)
print ('Quốc gia có sản lượng cà chua lớn nhất trong thời kì 1961-2000 là : ',␣
 ↪temp.iloc[0]['Entity'])
```

```
                    Entity        Yields
131            Netherlands  18773.273552
52                 Denmark  16433.734818
206         United Kingdom  15035.964096
142                 Norway  14902.662529
70                 Finland  13949.978889
..                     …           …
185                  Sudan    121.201797
195                  Timor    120.913399
167  Serbia and Montenegro    119.856898
23                  Bhutan     52.912198
174                Somalia     44.106899

[220 rows x 2 columns]
```

Quốc gia có sản lượng cà chua lớn nhất trong thời kì 1961-2000 là : Netherlands

[58]:
```
df['Entity'].unique()
```

[58]:
```
array(['Netherlands', 'Denmark', 'United Kingdom', 'Sweden', 'Norway',
       'Finland', 'Ireland', 'Belgium', 'Iceland', 'Western Europe (FAO)',
       'Austria', 'Germany', 'Switzerland', 'France',
       'Northern Europe (FAO)', 'New Zealand', 'United Arab Emirates',
       'Israel', 'Cyprus', 'Palestine', 'United States',
       'Northern America (FAO)', 'Kuwait', 'Portugal', 'Luxembourg',
       'High-income countries', 'Spain', 'Japan', 'Chile', 'South Korea',
       'Italy', 'Brazil', 'Oceania (FAO)', 'Oceania', 'Canada', 'Lebanon',
       'Southern Europe (FAO)', 'Australia', 'Estonia', 'Greece',
       'European Union (27)', 'European Union (27) (FAO)',
       'North America', 'Oman', 'Americas (FAO)', 'Jordan', 'Puerto Rico',
       'Turkey', 'Malta', 'South America (FAO)', 'South America',
       'Morocco', 'Tunisia', 'Syria', 'Argentina', 'South Africa',
       'Western Asia (FAO)', 'Egypt', 'Paraguay', 'Hungary',
       'Eastern Asia (FAO)', 'Southern Africa (FAO)', 'China',
       'China (FAO)', 'Croatia', 'Peru', 'Turkmenistan', 'Qatar',
       'Europe', 'Europe (FAO)', 'Northern Africa (FAO)', 'Albania',
       'French Guiana', 'Taiwan', 'World',
       'Upper-middle-income countries', 'Dominican Republic',
       'Cook Islands', 'Asia (FAO)', 'Asia', 'Iran', 'Guatemala',
       'Costa Rica', 'Armenia', 'El Salvador', 'Saudi Arabia', 'Thailand',
       'Bahrain', 'Net Food Importing Developing Countries (FAO)',
       'Belize', 'Reunion', 'Colombia', 'Mexico', 'Central America (FAO)',
       'Slovenia', 'Ecuador', 'Martinique', 'Algeria', 'Slovakia',
       'Uzbekistan', 'North Macedonia', 'Polynesia', 'Venezuela',
       'French Polynesia', 'Malaysia', 'Central Asia (FAO)', 'Africa',
       'Africa (FAO)', 'Tonga', 'Uruguay', 'Azerbaijan',
       'Southern Asia (FAO)', 'Jamaica', 'Lower-middle-income countries',
       'Cameroon', 'Kazakhstan', 'Kenya',
       'Land Locked Developing Countries (FAO)', 'Belarus', 'Kyrgyzstan',
       'India', 'Barbados', 'Caribbean (FAO)', 'Guadeloupe',
       'Small Island Developing States (FAO)', 'Czechia', 'Haiti',
       'Hong Kong', 'Low-income countries', 'Mali', 'Yemen',
       'Low Income Food Deficit Countries (FAO)', 'Poland', 'Georgia',
       'Nicaragua', 'Panama', 'Bulgaria', 'Middle Africa (FAO)',
       'South-eastern Asia (FAO)', 'Libya', 'Romania', 'Bolivia',
       'Indonesia', 'Cuba', 'Brunei', 'Iraq', 'Sudan (former)',
       'Eswatini', 'Trinidad and Tobago', 'Mauritius', 'Suriname',
       'Eastern Europe (FAO)', 'Senegal', 'Russia', 'Tajikistan',
       'Honduras', 'Saint Kitts and Nevis', 'Eastern Africa (FAO)',
       'Latvia', 'Dominica', 'Zambia', 'Burkina Faso', 'Sierra Leone',
       "Cote d'Ivoire", 'Cape Verde', 'Least Developed Countries (FAO)',
       'Ukraine', 'Niger', 'Pakistan', 'Ethiopia', 'Madagascar',
```

```
 'Comoros', 'Bahamas', 'Antigua and Barbuda', 'Fiji', 'Philippines',
 'Malawi', 'Liberia', 'Gabon', 'Moldova', 'Serbia and Montenegro',
 'Melanesia', 'North Korea', 'Tanzania', 'Sri Lanka', 'Grenada',
 'Mozambique', 'Democratic Republic of Congo', 'Rwanda',
 'Bangladesh', 'Uganda', 'Zimbabwe', 'Bosnia and Herzegovina',
 'Namibia', 'Western Africa (FAO)', 'Seychelles', 'Guyana',
 'Nigeria', 'Lithuania', 'Ghana', 'Benin', 'Congo',
 'Papua New Guinea', 'Togo', 'Angola', 'Timor', 'Somalia'],
dtype=object)
```

[59]:
```
DNA = ['Timor', 'Indonesia', 'Malaysia', 'Brunei', 'Philippines']
mask = tomato['Entity'].isin(DNA)
tomato[mask]
```

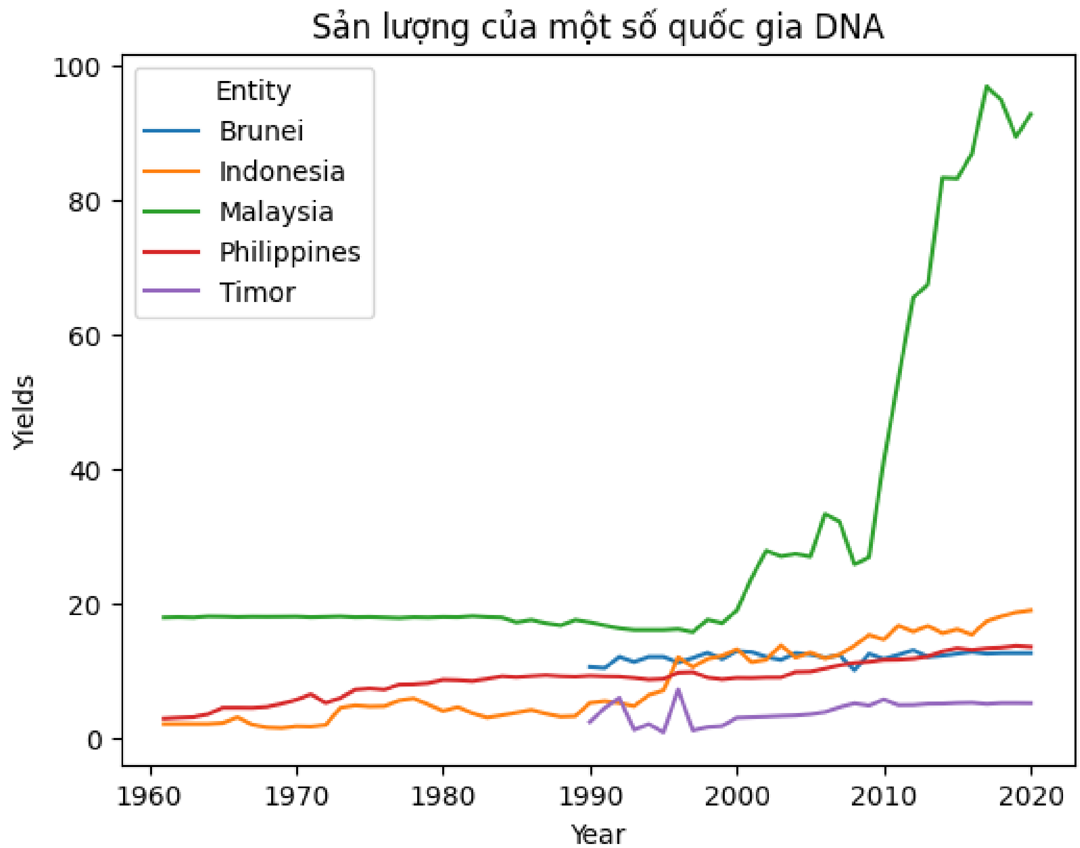
[59]:

	Entity	Code	Year	Yields
1372	Brunei	BRN	1990	10.5000
1373	Brunei	BRN	1991	10.3333
1374	Brunei	BRN	1992	12.0000
1375	Brunei	BRN	1993	11.2500
1376	Brunei	BRN	1994	12.0000
…	…	…	…	…
10002	Timor	TLS	2016	5.1818
10003	Timor	TLS	2017	4.9821
10004	Timor	TLS	2018	5.1182
10005	Timor	TLS	2019	5.1182
10006	Timor	TLS	2020	5.1000

[242 rows x 4 columns]

[101]:
```
import seaborn as sns
import matplotlib as plt
sns.lineplot(data=tomato[mask], x="Year", y="Yields", hue="Entity").set(title =␣
↪"Sản lượng của một số quốc gia DNA")
# plt.title ('Sản lượng của một số quốc gia DNA')
```

[101]: [Text(0.5, 1.0, 'Sản lượng của một số quốc gia DNA')]



0.1 Phan 2

[19]:
```
df2 = pd.read_csv('/Users/thutranghoa/Code/Data_analysis/Data/
↪1667260416_774__Marketing.csv')
df2
```

[19]:
```
        Age  Gender OwnHome  Married Location  Salary  Children History  \
0       Old  Female     Own   Single      Far   47500         0    High
1    Middle    Male    Rent   Single    Close   63600         0    High
2     Young  Female    Rent   Single    Close   13500         0     Low
3    Middle    Male     Own  Married    Close   85600         1    High
4    Middle  Female     Own   Single    Close   68400         0    High
..      …       …       …       …       …       …        …
995   Young  Female    Rent   Single    Close   19400         1     NaN
996  Middle    Male    Rent   Single      Far   40500         1     NaN
997     Old    Male     Own   Single    Close   44800         0  Medium
998  Middle    Male     Own  Married    Close   79000         2  Medium
999   Young    Male    Rent  Married    Close   53600         1  Medium
```

```
     Catalogs  AmountSpent
0           6          755
1           6         1318
2          18          296
3          18         2436
4          12         1304
..        …           …
995        18          384
996        18         1073
997        24         1417
998        18          671
999        24          973

[1000 rows x 10 columns]
```

[21]:
```
df2 = df2.dropna()
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 697 entries, 0 to 999
Data columns (total 10 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Age          697 non-null    object
 1   Gender       697 non-null    object
 2   OwnHome      697 non-null    object
 3   Married      697 non-null    object
 4   Location     697 non-null    object
 5   Salary       697 non-null    int64
 6   Children     697 non-null    int64
 7   History      697 non-null    object
 8   Catalogs     697 non-null    int64
 9   AmountSpent  697 non-null    int64
dtypes: int64(4), object(6)
memory usage: 59.9+ KB
```

[26]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

df2['Age'] = le.fit_transform(df2['Age'])
```

```
/var/folders/cs/8r3m5sjs0rd7ts526sxtp81c0000gn/T/ipykernel_20764/4019742341.py:4
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

```
  df2['Age'] = le.fit_transform(df2['Age'])
```

[28]:
```
df2['Gender'] = le.fit_transform(df2['Gender'])
df2['OwnHome'] = le.fit_transform(df2['OwnHome'])
df2['Married'] = le.fit_transform(df2['Married'])
df2['Location'] = le.fit_transform(df2['Location'])
df2['History'] = le.fit_transform(df2['History'])
```

```
/var/folders/cs/8r3m5sjs0rd7ts526sxtp81c0000gn/T/ipykernel_20764/3487257344.py:1
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df2['Gender'] = le.fit_transform(df2['Gender'])
/var/folders/cs/8r3m5sjs0rd7ts526sxtp81c0000gn/T/ipykernel_20764/3487257344.py:2
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df2['OwnHome'] = le.fit_transform(df2['OwnHome'])
/var/folders/cs/8r3m5sjs0rd7ts526sxtp81c0000gn/T/ipykernel_20764/3487257344.py:3
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df2['Married'] = le.fit_transform(df2['Married'])
/var/folders/cs/8r3m5sjs0rd7ts526sxtp81c0000gn/T/ipykernel_20764/3487257344.py:4
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df2['Location'] = le.fit_transform(df2['Location'])
/var/folders/cs/8r3m5sjs0rd7ts526sxtp81c0000gn/T/ipykernel_20764/3487257344.py:5
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df2['History'] = le.fit_transform(df2['History'])
```

[29]:
```
df2
```

[29]:
```
     Age  Gender  OwnHome  Married  Location  Salary  Children  History  \
0      1       0        0        1         1   47500         0        0
1      0       1        1        1         0   63600         0        0
2      2       0        1        1         0   13500         0        1
3      0       1        0        0         0   85600         1        0
4      0       0        0        1         0   68400         0        0
..   ...     ...      ...      ...       ...     ...       ...      ...
991    1       0        1        1         1   11700         0        1
993    0       0        0        0         1   99200         0        0
997    1       1        0        1         0   44800         0        2
998    0       1        0        0         0   79000         2        2
999    2       1        1        0         0   53600         1        2

     Catalogs  AmountSpent
0           6          755
1           6         1318
2          18          296
3          18         2436
4          12         1304
..        ...          ...
991        18          540
993        24         5503
997        24         1417
998        18          671
999        24          973

[697 rows x 10 columns]
```

[37]:
```
X = df2.drop(['AmountSpent'], axis=1)
y = df2['AmountSpent']

print (X.shape)
print (y.shape)
```

```
(697, 9)
(697,)
```

[34]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
↪random_state=44)
```

[35]:
```
X_train
```

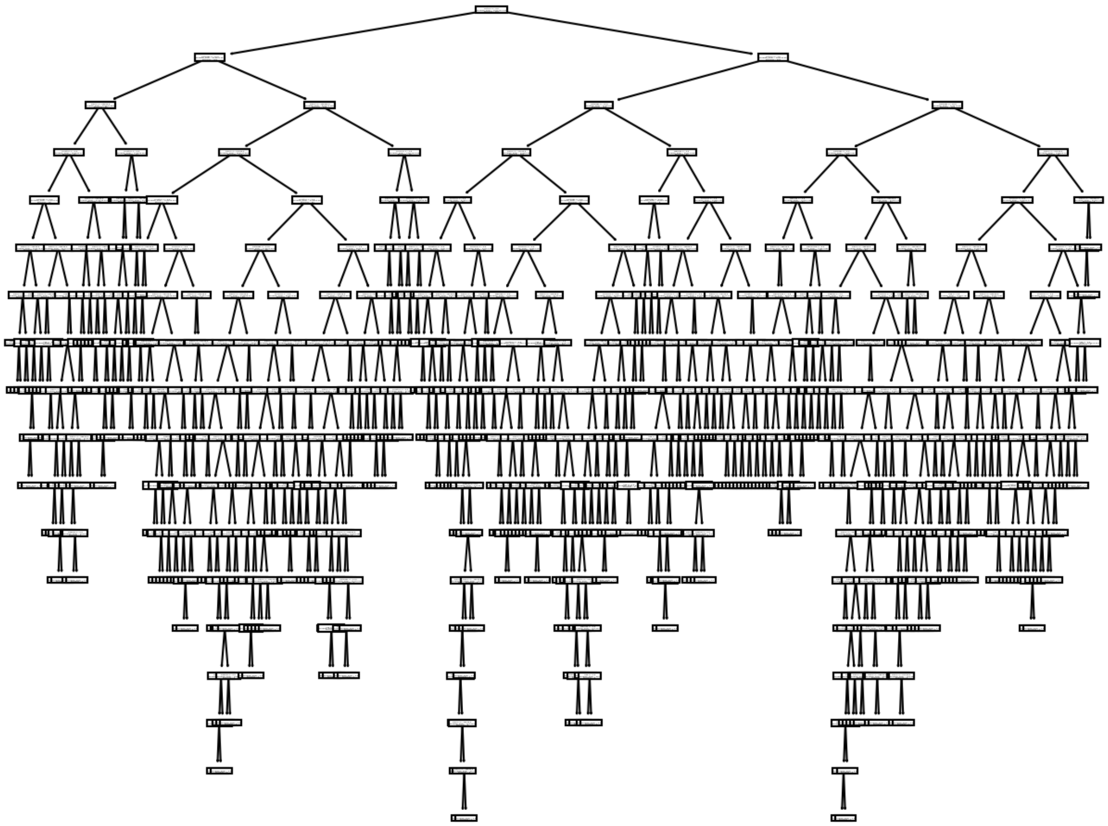
[35]:
```
     Age  Gender  OwnHome  Married  Location  Salary  Children  History  \
345    2       1        0        1         0   31900         0        2
973    0       0        0        1         1   56200         0        0
```

```
230     1        0        1        1        1   15000        0        1
907     0        1        1        0        1   52800        2        2
960     1        1        0        1        0   71300        0        0
..    …      …       …       …       …       …       …        …
116     1        1        0        0        0   58300        0        2
138     2        0        0        0        1   68000        0        0
829     0        1        0        0        1  130600        0        0
245     0        1        0        0        1  120000        3        0
601     2        1        1        1        0   19800        2        1

     Catalogs
345         6
973        18
230        12
907        12
960        18
..        …
116        24
138        24
829        18
245        18
601        12

[487 rows x 9 columns]
```

##Decision Tree Regressor

[63]:
```python
# Import the necessary modules and libraries
import matplotlib.pyplot as plt
import numpy as np

from sklearn.tree import DecisionTreeRegressor

model = DecisionTreeRegressor(random_state=44)
model.fit(X_train, y_train)
predictions = model.predict(X_test)
```

[64]:
```python
from sklearn.tree import plot_tree
plt.figure(figsize=(10,8), dpi=150)
plot_tree(model, feature_names=X.columns);
```



[66]:
```
from sklearn.metrics import mean_squared_error , r2_score
print ('MSE of Decision Tree = ', mean_squared_error(y_test, predictions))
print ('R2_score of Decision Tree = ', r2_score(y_test, predictions))
```

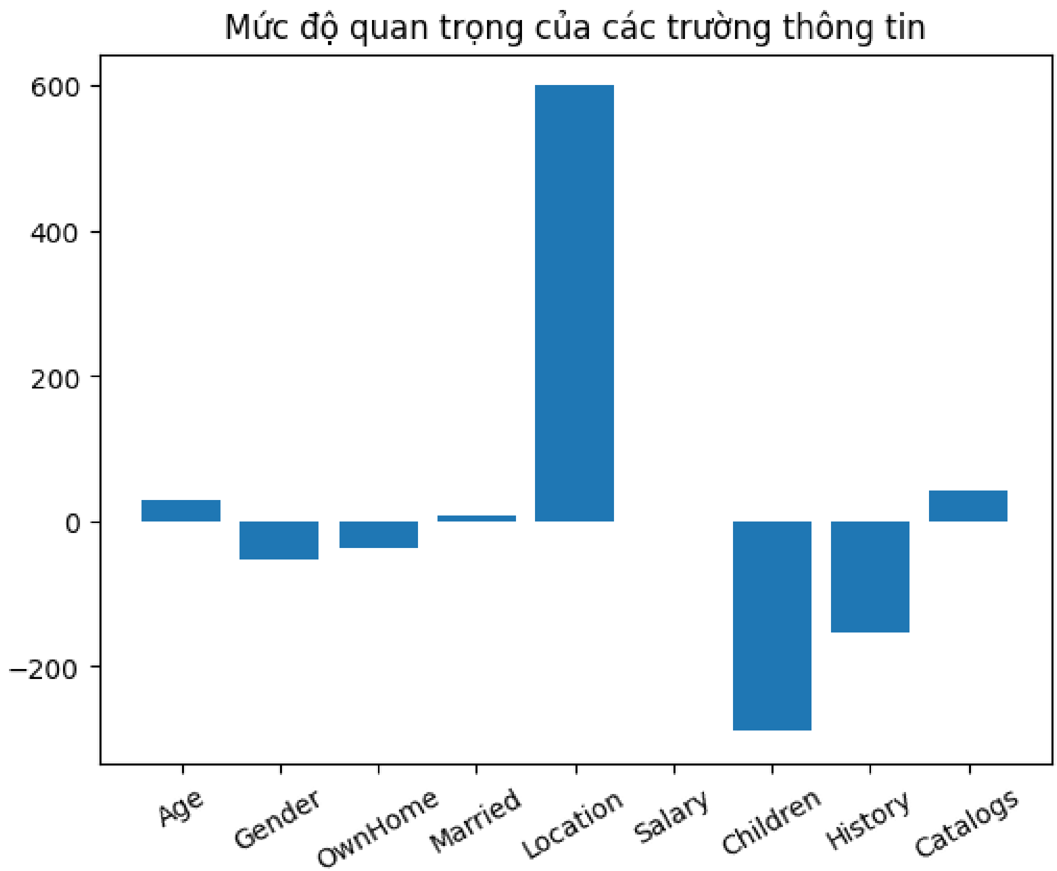
```
MSE of Decision Tree =  338484.38095238095
R2_score of Decision Tree =  0.6578431761813559
```

##Linear Regression

[75]:
```
from sklearn.linear_model import LinearRegression

LR = LinearRegression()
LR.fit(X_train, y_train)
predictions = LR.predict(X_test)

print ('MSE of LinearRegression= ', mean_squared_error(y_test, predictions))
print ('R2_score of Linear Regression= ', r2_score(y_test, predictions))
```

```
MSE of LinearRegression=  246144.84279594294
R2_score of Linear Regression=  0.7511845675908819
```

[95]:
```python
A = list(df2.columns)
A.remove('AmountSpent')
importance = LR.coef_

# plot feature importance
feature = df2.columns
plt.bar(A, importance)
plt.title ('Mức độ quan trọng của các trường thông tin')
plt.xticks(rotation = 30)
plt.show()
```



##SVM

[72]:
```python
from sklearn.svm import SVR
svr = SVR(kernel="linear")
svr.fit(X_train, y_train)

predictions = svr.predict(X_test)
```

```
print ('MSE of SVR = ', mean_squared_error(y_test, predictions))
print ('R2_score of SVR = ', r2_score(y_test, predictions))
```

```
MSE of SVR =  1043035.9377248775
R2_score of SVR =  -0.0543525304668413
```

[73]:
```
from sklearn.svm import SVR
svr = SVR(kernel="poly")
svr.fit(X_train, y_train)

predictions = svr.predict(X_test)

print ('MSE of SVR = ', mean_squared_error(y_test, predictions))
print ('R2_score of SVR = ', r2_score(y_test, predictions))
```

```
MSE of SVR =  711649.5007908453
R2_score of SVR =  0.2806293390034591
```

[74]:
```
from sklearn.svm import SVR
svr = SVR(kernel="rbf")
svr.fit(X_train, y_train)

predictions = svr.predict(X_test)

print ('MSE of SVR = ', mean_squared_error(y_test, predictions))
print ('R2_score of SVR = ', r2_score(y_test, predictions))
```

```
MSE of SVR =  962101.6435287277
R2_score of SVR =  0.027459873881748642
```

0.2 Nhận xét

Trong 3 model sử dụng thì Linear Regression có R2_score cao nhất, chứng tỏ model này phù hợp nhất. Trong đó, thuôc tính ‘Location’ có ảnh hưởng lớn nhất đến khoản tiền chi tiêu