Supermarket_notebook

July 11, 2023

1 Final Project

Welcome to the final practical project for our course on Data Science Bootcamp. Throughout this project, you will go through the entire data science process, starting from data loading and cleaning, all the way to running a model and making predictions. This hands-on project will provide you with valuable experience and allow you to apply the concepts and techniques you've learned in the course. Get ready to dive into real-world data analysis and build your skills as a data scientist!

1.1 Important Remarks:

- The ultimate goal of this project is to conduct comprehensive data analysis and build 2 models using the provided datasets.
- Code is not the only thing graded here. Well-written and understandable documentation of your code is to be expected
- Clear reasoning behind your choices in every step of the notebook is important. Be it the choice of a data cleaning technique or selecting certain features in your analysis or the choice of your 2 models.

2 Importing packages

[55]:
```
# Import numpy and pandas
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split
```

3 Load the dataset into data

[56]:
```
# Load the dataset
df = pd.read_csv('supermarket_survey.csv', sep=';', header=0)
```

4 Dataset overview and statistical summary

[57]:
```
#get information about datset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 353 entries, 0 to 352
Data columns (total 46 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   randomInt               353 non-null    int64
 1   age                     345 non-null    object
 2   gender                  347 non-null    object
 3   district                334 non-null    object
 4   modeOfTransportation    341 non-null    object
 5   distance                338 non-null    object
 6   G03Q13amountOfPeople    345 non-null    object
 7   income                  331 non-null    float64
 8   frequency               339 non-null    object
 9   days[1]                 353 non-null    object
 10  days[2]                 353 non-null    object
 11  days[3]                 353 non-null    object
 12  days[4]                 353 non-null    object
 13  days[5]                 353 non-null    object
 14  days[6]                 353 non-null    object
 15  days[7]                 353 non-null    object
 16  time[1]                 353 non-null    object
 17  time[2]                 353 non-null    object
 18  time[3]                 353 non-null    object
 19  time[4]                 353 non-null    object
 20  time[5]                 353 non-null    object
 21  moneySpent              338 non-null    object
 22  orderingItems           334 non-null    object
 23  deliveringItems         333 non-null    object
 24  willingPayDelivery      166 non-null    object
 25  findProducts            334 non-null    object
 26  usingDiscounts          326 non-null    object
 27  preferCash              331 non-null    object
 28  preferCashless          329 non-null    object
 29  isRelaxing              327 non-null    object
 30  satisGeneralStore       332 non-null    float64
 31  satisMusic              288 non-null    float64
```

```
 32  satisQualityProducts     329 non-null    float64
 33  satisGeneralAssortment   330 non-null    float64
 34  satisVeganProducts       274 non-null    float64
 35  satisOrganicProducts     301 non-null    float64
 36  satisGlutenfreeProducts  209 non-null    float64
 37  satisAnimalProducts      307 non-null    float64
 38  ideasExtendedBusiness    324 non-null    float64
 39  ideasHelpCarry           322 non-null    float64
 40  ideasCustomerCouncil     318 non-null    float64
 41  ideasFreeWifi            324 non-null    float64
 42  ideasTouchDisplay        320 non-null    float64
 43  ideasSelfCheckout        323 non-null    float64
 44  ideasBikeParking         312 non-null    float64
 45  ideasUndergroundParking  300 non-null    float64
dtypes: float64(17), int64(1), object(28)
memory usage: 127.0+ KB
```

[58]:

```
#disply 100 colunms
df.head(100)
```

[58]:

```
    randomInt    age  gender    district modeOfTransportation  \
0           4    NaN    Male      Godham              Own Car
1           4    NaN     NaN         NaN                  NaN
2           3  20-25  Female  Springtown              Own Car
3           4    NaN     NaN         NaN                  NaN
4           3  15-20    Male   Piltunder              Own Car
..        …      …      …         …                   …
95          3  55-60    Male         NaN              Own Car
96          2  60-65    Male      Godham              Walking
97          4    >75    Male    Duckborg              Own Car
98          4  45-50  Female         NaN                  NaN
99          3  35-40    Male      Godham              Walking

                     distance G03Q13amountOfPeople    income    frequency  \
0                       1-2km                    3  120000.0        Twice
1                         NaN                  NaN       NaN          NaN
2                        >7km                    2      15.0  Three times
3                         NaN                  NaN    1337.0          NaN
4                       1-2km                    4  250000.0        Twice
..                        …                    …        …          …
95                      3-5km            5 or more    3500.0        Twice
96          500 meters to 1km                    2     200.0         Once
97                      5-7km                    2  200000.0         Once
98                      1-2km                    3    4000.0         Once
99  Less than few hundred meters                 1   15000.0  Three times

   days[1]  … satisGlutenfreeProducts satisAnimalProducts  \
```

```
0       No  …                   8.0                7.0
1       No  …                   NaN                NaN
2       No  …                   7.0                NaN
3       No  …                   NaN                NaN
4       No  …                   8.0                1.0
..     …  …                   …                  …
95      No  …                   NaN                8.0
96      No  …                   6.0                9.0
97      No  …                   NaN                9.0
98     Yes  …                   NaN                9.0
99      No  …                   NaN                8.0

   ideasExtendedBusiness ideasHelpCarry ideasCustomerCouncil ideasFreeWifi  \
0                    2.0            4.0                  3.0           4.0
1                    NaN            NaN                  NaN           NaN
2                    7.0            7.0                  7.0           7.0
3                    NaN            NaN                  NaN           NaN
4                    9.0            2.0                  1.0          10.0
..                   …             …                   …            …
95                   9.0            8.0                  8.0           NaN
96                   8.0           10.0                 10.0          10.0
97                  10.0            5.0                  1.0           1.0
98                   NaN            1.0                  1.0           5.0
99                   9.0            1.0                  1.0           1.0

   ideasTouchDisplay ideasSelfCheckout ideasBikeParking  \
0                NaN               4.0              NaN
1                NaN               NaN              NaN
2                NaN               7.0              7.0
3                NaN               NaN              NaN
4               10.0              10.0              8.0
..               …                 …               …
95               NaN               8.0              8.0
96              10.0              10.0             10.0
97               2.0              10.0              1.0
98               7.0              10.0              NaN
99               1.0               5.0              7.0

   ideasUndergroundParking
0                      NaN
1                      NaN
2                      7.0
3                      NaN
4                      NaN
..                     …
95                     NaN
96                     7.0
```

```
97                       1.0
98                       NaN
99                       1.0

[100 rows x 46 columns]
```

[59]:
```
#statistical summary of the numerical columns in the DataFrame.
#which calculate various descriptive statistics, such as count, mean, standard␣
↪deviation, minimum, quartiles, and maximum, for each numerical column.
df.describe()
```

[59]:
```
         randomInt         income  satisGeneralStore  satisMusic  \
count   353.000000     331.000000         332.000000  288.000000
mean      2.609065   66275.568882           7.424699    5.236111
std       1.105322  132542.950482           1.705790    2.507094
min       1.000000  -99932.000000           1.000000    1.000000
25%       2.000000    2290.000000           7.000000    3.000000
50%       3.000000   21000.000000           8.000000    5.000000
75%       4.000000   80284.000000           8.000000    7.000000
max       4.000000  999999.000000          10.000000   10.000000

       satisQualityProducts  satisGeneralAssortment  satisVeganProducts  \
count            329.000000              330.000000          274.000000
mean               7.498480                7.278788            6.350365
std                1.479792                1.674366            2.177444
min                1.000000                1.000000            1.000000
25%                7.000000                7.000000            5.000000
50%                8.000000                8.000000            7.000000
75%                8.000000                8.000000            8.000000
max               10.000000               10.000000           10.000000

       satisOrganicProducts  satisGlutenfreeProducts  satisAnimalProducts  \
count            301.000000               209.000000           307.000000
mean               6.767442                 6.315789             7.348534
std                1.981347                 2.269317             1.902618
min                1.000000                 1.000000             1.000000
25%                6.000000                 5.000000             6.500000
50%                7.000000                 6.000000             8.000000
75%                8.000000                 8.000000             9.000000
max               10.000000                10.000000            10.000000

       ideasExtendedBusiness  ideasHelpCarry  ideasCustomerCouncil  \
count             324.000000      322.000000            318.000000
mean                6.919753        3.711180              3.232704
std                 3.129760        3.027465              2.668179
min                 1.000000        1.000000              1.000000
25%                 5.000000        1.000000              1.000000
```

```
50%                  8.000000         2.000000                2.000000
75%                 10.000000         6.000000                5.000000
max                 10.000000        10.000000               10.000000

       ideasFreeWifi  ideasTouchDisplay  ideasSelfCheckout  ideasBikeParking  \
count     324.000000         320.000000         323.000000        312.000000
mean        6.410494           5.571875           7.857585          7.602564
std         3.147757           3.197936           2.668804          2.752793
min         1.000000           1.000000           1.000000          1.000000
25%         4.000000           3.000000           7.000000          6.000000
50%         7.000000           6.000000           9.000000          8.000000
75%         9.000000           9.000000          10.000000         10.000000
max        10.000000          10.000000          10.000000         10.000000

       ideasUndergroundParking
count               300.000000
mean                  5.396667
std                   3.321057
min                   1.000000
25%                   2.000000
50%                   6.000000
75%                   8.000000
max                  10.000000
```

[60]:
```
df.gender.value_counts()
```

[60]:
```
Male                 232
Female                84
Prefer not to say     22
Diverse                9
Name: gender, dtype: int64
```

[61]:
```
df.moneySpent.value_counts()
```

[61]:
```
Between 25 and 50 USD     103
Between 50 and 75          58
Less than 25 USD           58
More than 125 USD          44
Between 75 and 100 USD     42
100 to 125 USD             33
Name: moneySpent, dtype: int64
```

[62]:
```
df.orderingItems.value_counts()
```

[62]:
```
…selecting them myself in the store.    250
… ordering online.                       84
Name: orderingItems, dtype: int64
```

[63]:
```
df.orderingItems.value_counts()
```

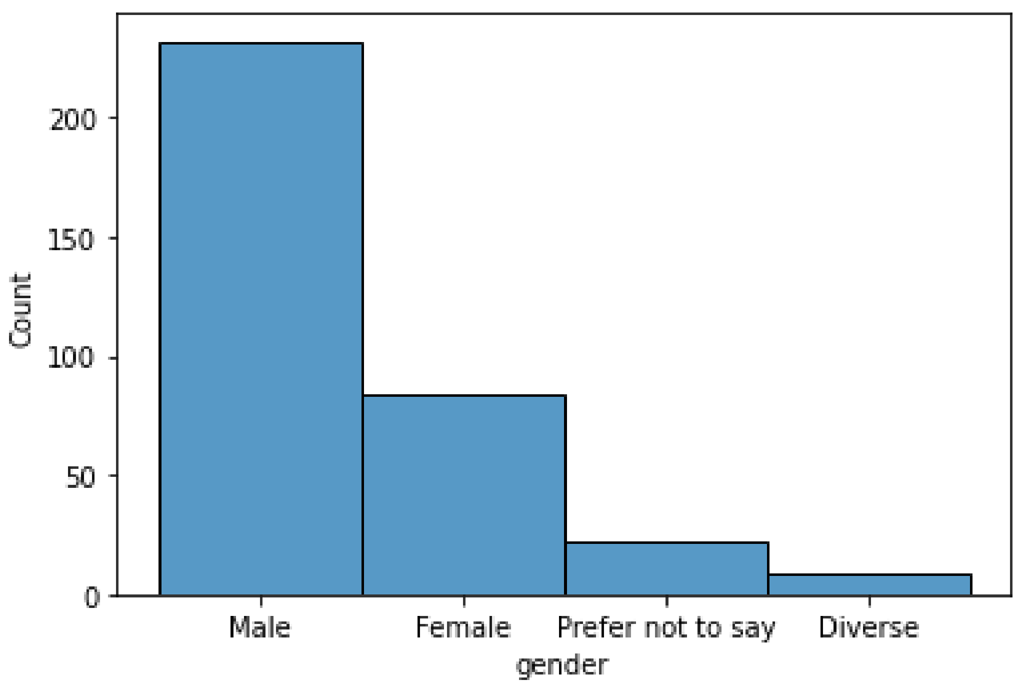
[63]:
```
…selecting them myself in the store.    250
… ordering online.                       84
Name: orderingItems, dtype: int64
```

[64]:
```
df.satisQualityProducts.value_counts()
```

[64]:
```
8.0     103
7.0      84
9.0      60
6.0      30
5.0      21
10.0     19
4.0       7
3.0       3
2.0       1
1.0       1
Name: satisQualityProducts, dtype: int64
```

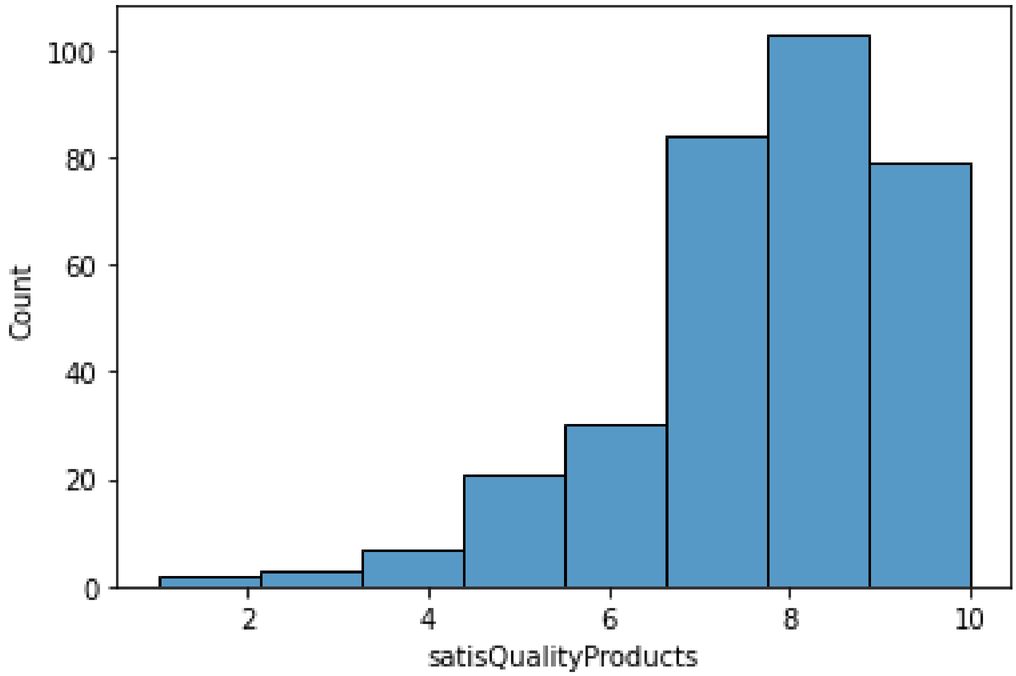
[65]:
```
sns.histplot(x=df["gender"], bins=5)
```

[65]: <AxesSubplot:xlabel='gender', ylabel='Count'>



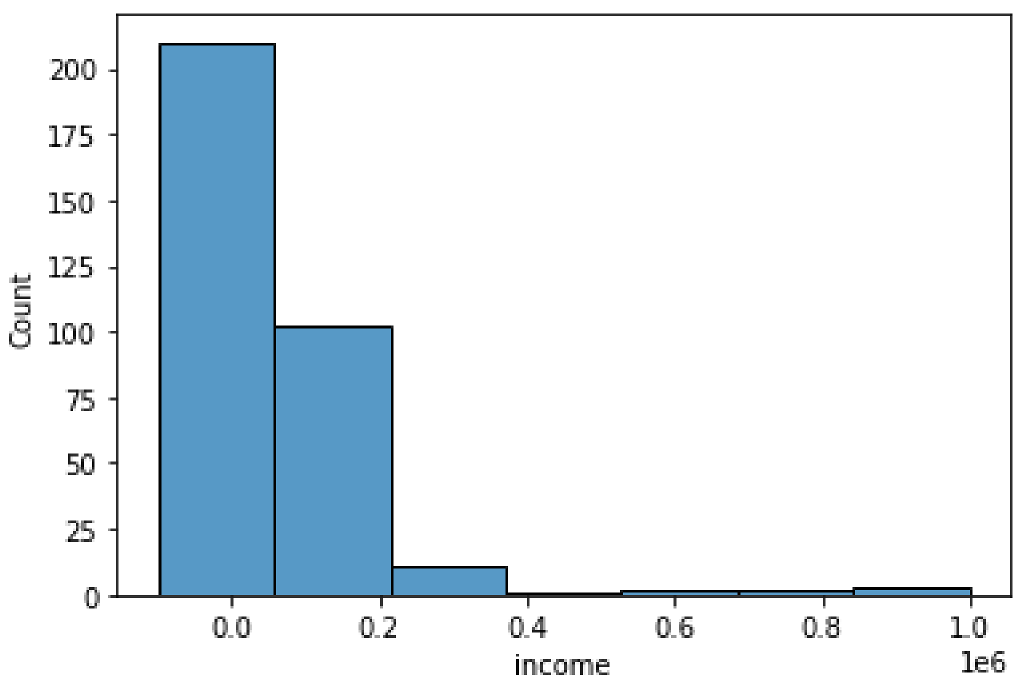
[66]:
```
sns.histplot(x=df["satisQualityProducts"], bins=8)
```

[66]: <AxesSubplot:xlabel='satisQualityProducts', ylabel='Count'>



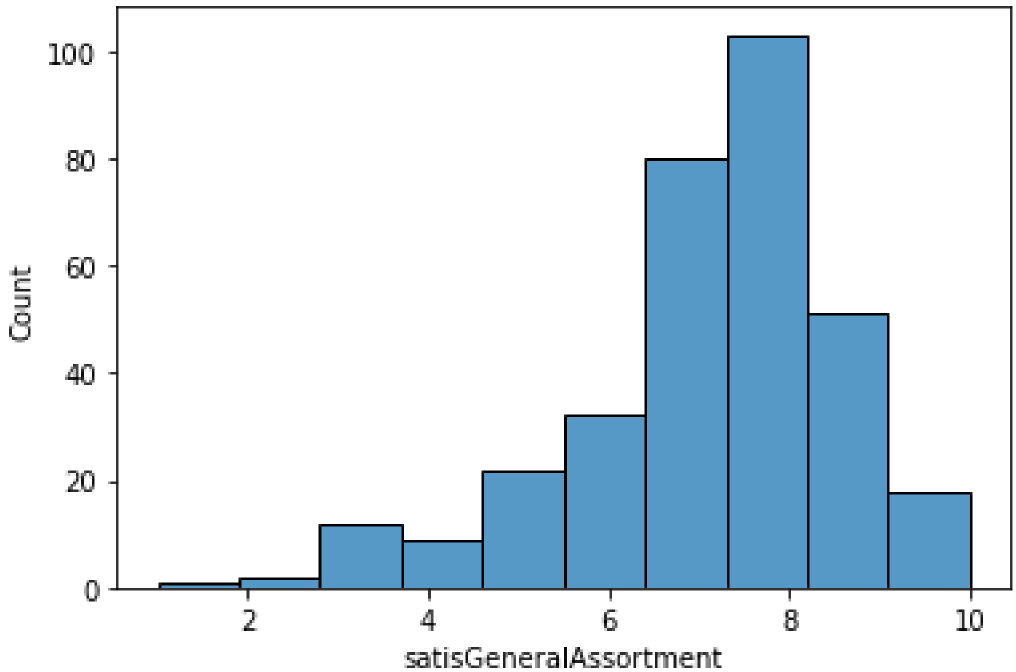

```
[67]: sns.histplot(x=df["income"], bins=7)
```

[67]: <AxesSubplot:xlabel='income', ylabel='Count'>



[68]:
```
sns.histplot(data = df["satisGeneralAssortment"], bins=10)
```

[68]: <AxesSubplot:xlabel='satisGeneralAssortment', ylabel='Count'>



[69]:
```
# have overviow about income
df.income.describe()
```

[69]:
```
count       331.000000
mean      66275.568882
std      132542.950482
min      -99932.000000
25%        2290.000000
50%       21000.000000
75%       80284.000000
max      999999.000000
Name: income, dtype: float64
```

[70]:
```
#detectime missing income data
df.income.isna().sum()
```

[70]: 22

[71]:
```
# detect negative income
negative_income= df[df["income"]<0]
negative_income
```

[71]:
```
     randomInt    age              gender    district modeOfTransportation  \
75           3  15-20  Prefer not to say  Metrapalis              Bicycle
190          3  20-25               Male     Godham              Bicycle

               distance G03Q13amountOfPeople   income            frequency  \
75   500 meters to 1km            5 or more -99932.0                  NaN
190             1-2km                     1 -10000.0  More than four times

    days[1]  … satisGlutenfreeProducts satisAnimalProducts  \
75       No  …                    10.0                 9.0
190     Yes  …                     1.0                 NaN

    ideasExtendedBusiness ideasHelpCarry ideasCustomerCouncil ideasFreeWifi  \
75                    1.0            1.0                  1.0          10.0
190                  10.0            1.0                  1.0           1.0

    ideasTouchDisplay ideasSelfCheckout ideasBikeParking  \
75               10.0              10.0             10.0
190               1.0               1.0             10.0

    ideasUndergroundParking
75                      1.0
190                     1.0

[2 rows x 46 columns]
```

[72]:
```
#detect users with income higher than 10000
montly_income= df[df["income"]<10000]
montly_income
```

[72]:
```
     randomInt    age              gender    district   modeOfTransportation  \
2            3  20-25             Female  Springtown                Own Car
3            4    NaN                NaN         NaN                    NaN
5            3  20-25  Prefer not to say  Metrapalis                Walking
6            2  60-65               Male      Godham                Own Car
8            1  25-30               Male  Metrapalis                Walking
..         …      …                 …         …                    …
345          1  25-30               Male  Metrapalis  Public transportation
349          2  30-35               Male      Godham                Own Car
350          3  20-25               Male      Godham                Walking
351          4  35-40               Male    Piltunder                Own Car
352          4  45-50               Male     Duckborg                Own Car
```

```
          distance G03Q13amountOfPeople  income    frequency days[1]  …  \
2             >7km                  2    15.0  Three times      No  …
3              NaN                NaN  1337.0          NaN      No  …
5   500 meters to 1km                  1   500.0        Twice      No  …
6           1-2km                  2  5000.0         Once      No  …
8   500 meters to 1km                  1   600.0   Four times     Yes  …
..             …                  …     …          …     …  …
345         3-5km                  1  1000.0        Twice     Yes  …
349         1-2km                  3    50.0  Three times      No  …
350         1-2km                  2     5.5         Once      No  …
351         3-5km                  4   600.0         Once      No  …
352          >7km                  1  2500.0         Once      No  …

    satisGlutenfreeProducts satisAnimalProducts ideasExtendedBusiness  \
2                       7.0                 NaN                   7.0
3                       NaN                 NaN                   NaN
5                      10.0                10.0                   9.0
6                       NaN                 7.0                   5.0
8                       6.0                 8.0                  10.0
..                      …                  …                     …
345                     9.0                 9.0                   9.0
349                     5.0                 7.0                   9.0
350                     7.0                 7.0                   4.0
351                     7.0                 8.0                   8.0
352                    10.0                10.0                  10.0

    ideasHelpCarry ideasCustomerCouncil ideasFreeWifi ideasTouchDisplay  \
2              7.0                  7.0           7.0               NaN
3              NaN                  NaN           NaN               NaN
5              1.0                  1.0           9.0               9.0
6              2.0                  2.0           6.0               3.0
8              2.0                  3.0           4.0               3.0
..             …                   …            …                 …
345            1.0                  5.0          10.0              10.0
349            8.0                  8.0           8.0               8.0
350            6.0                  NaN           8.0               9.0
351            9.0                  9.0          10.0              10.0
352           10.0                 10.0          10.0              10.0

    ideasSelfCheckout ideasBikeParking ideasUndergroundParking
2                 7.0              7.0                     7.0
3                 NaN              NaN                     NaN
5                10.0              1.0                     1.0
6                 9.0              9.0                     9.0
8                10.0             10.0                     2.0
..                …               …                      …
345              10.0             10.0                    10.0
```

```
349                9.0              9.0                        10.0
350                9.0              9.0                         8.0
351               10.0             10.0                         7.0
352               10.0             10.0                        10.0

[152 rows x 46 columns]
```

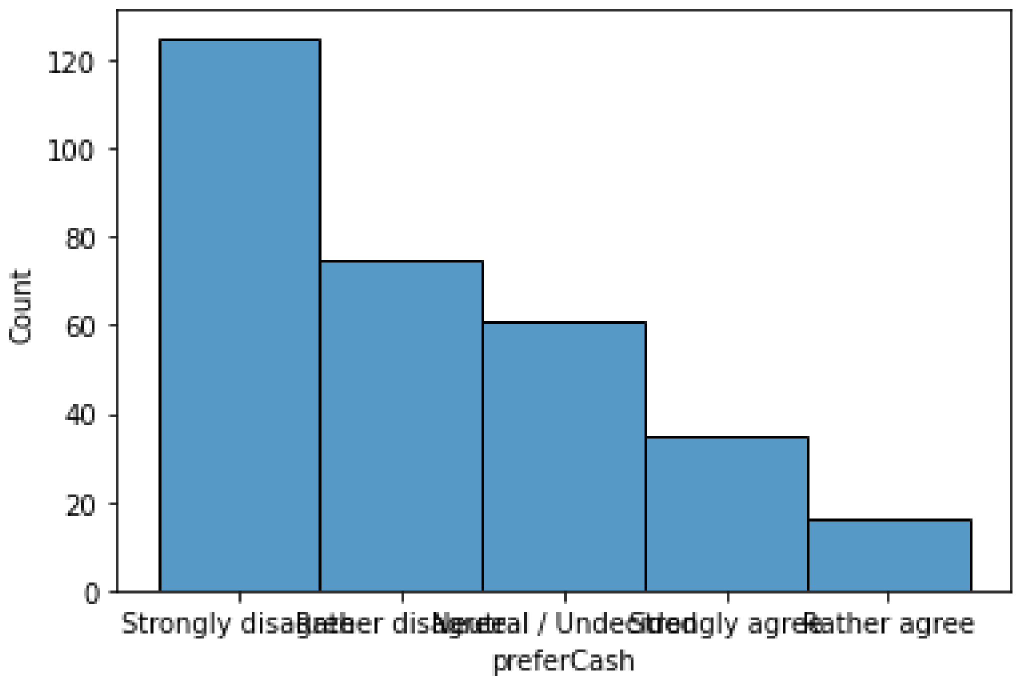
[73]:
```
#keep just positive income
df= df[df["income"]>0]
```

[74]:
```
df.income.describe()
```

[74]:
```
count       327.000000
mean      67422.462691
std      132862.089848
min           3.000000
25%        2500.000000
50%       24000.000000
75%       82784.000000
max      999999.000000
Name: income, dtype: float64
```

[75]:
```
#Csh vs cashless
sns.histplot(df.preferCash)
```

[75]: <AxesSubplot:xlabel='preferCash', ylabel='Count'>



[76]:
```
#count user who prefer cash
df.preferCash.value_counts()
```

[76]:
```
Strongly disagree      125
Rather disagree         75
Neutral / Undecided     61
Strongly agree          35
Rather agree            16
Name: preferCash, dtype: int64
```

[77]:
```
#count user who not prefer cash
df.preferCashless.value_counts()
```

[77]:
```
Strongly agree         171
Rather agree            52
Neutral / Undecided     48
Rather disagree         22
Strongly disagree       16
Name: preferCashless, dtype: int64
```

5 Data cleaning

[78]:
```
df = df.drop(['randomInt'], axis=1)
df = df.drop("willingPayDelivery", axis=1)
df = df.drop("distance", axis=1)
```

[79]:
```
days_columns = ['time[1]', 'time[2]', 'time[3]', 'time[4]', 'time[5]']
df.drop(days_columns, axis=1, inplace=True)
```

[80]:
```
days_columns = ['days[1]', 'days[2]', 'days[3]', 'days[4]', 'days[5]',␣
 ↪'days[6]', 'days[7]']
df.drop(days_columns, axis=1, inplace=True)
```

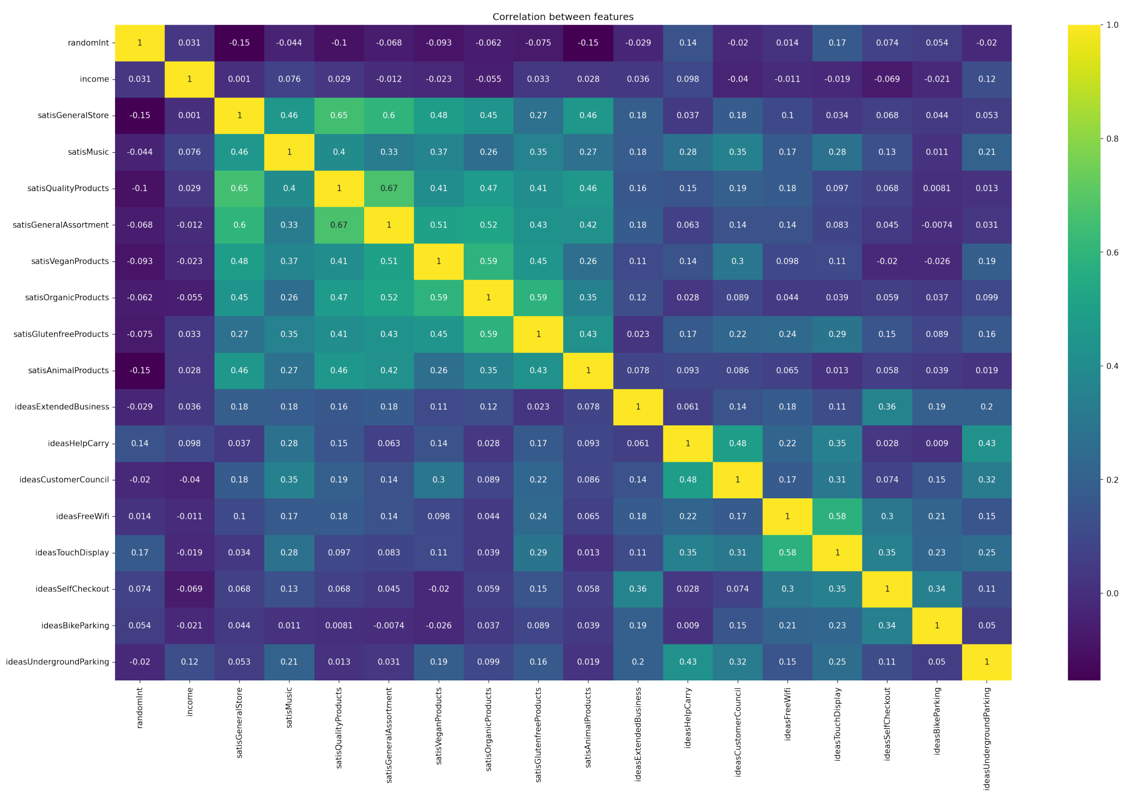
[81]:
```
# Handline with missing values
columns_with_missing_values = ['age', 'gender', 'district',␣
 ↪'modeOfTransportation',
                               'G03Q13amountOfPeople', 'income', 'frequency',␣
 ↪'moneySpent',
                               'orderingItems', 'deliveringItems',␣
 ↪'findProducts', 'usingDiscounts',
```

```
                                    'preferCash', 'preferCashless', 'isRelaxing',␣
 ↪'satisGeneralStore', 'satisMusic', 'satisQualityProducts',␣
 ↪'satisGeneralAssortment', 'satisVeganProducts', 'satisOrganicProducts',␣
 ↪'satisGlutenfreeProducts', 'satisAnimalProducts', 'ideasExtendedBusiness',␣
 ↪'ideasHelpCarry', 'ideasCustomerCouncil', 'ideasFreeWifi',␣
 ↪'ideasTouchDisplay', 'ideasSelfCheckout',
                                    'ideasBikeParking', 'ideasUndergroundParking']

# Fill missing values with appropriate method
df[columns_with_missing_values] = df[columns_with_missing_values].
 ↪fillna(df[columns_with_missing_values].mode().iloc[0])
```

6 EDA

[16]:
```
# Heatmap of the correlation of the features
plt.figure(figsize=(25,15),dpi=150)
sns.heatmap(df.corr(),cmap='viridis',annot=True)
plt.title('Correlation between features');
```

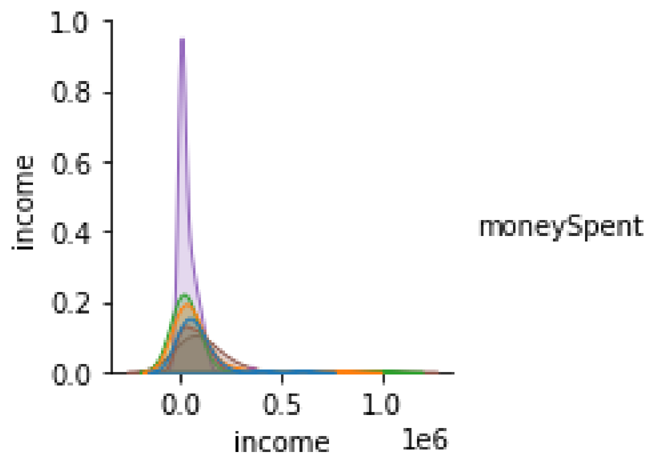


[17]:
```
# understanding the relations between variables
data = df[['orderingItems', 'age', 'gender', 'income', 'frequency',
```

```
            'deliveringItems', 'findProducts', 'usingDiscounts',
            'preferCash', 'preferCashless']].copy()

data['moneySpent'] = df['moneySpent']

sns.pairplot(data, hue='moneySpent')
plt.show()
```

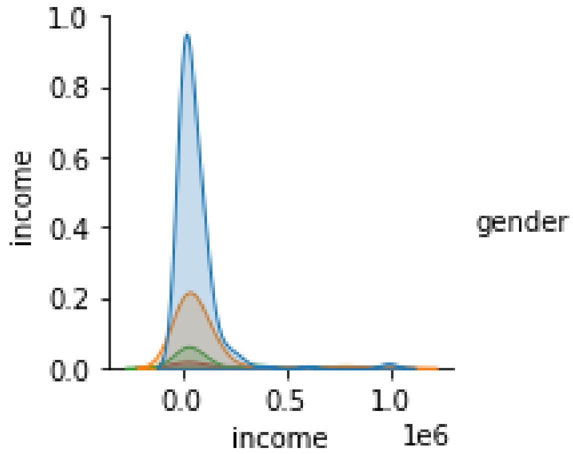


[19]:
```
# understanding the relations between variables
data = df[['orderingItems', 'age', 'moneySpent', 'income', 'frequency',
            'deliveringItems', 'findProducts', 'usingDiscounts',
            'preferCash', 'preferCashless']].copy()

data['gender'] = df['gender']

sns.pairplot(data, hue='gender')
```

[19]:
```
<seaborn.axisgrid.PairGrid at 0x7fe5bb7d82b0>
```



7 Data Processing and normalization

[82]:
```
# Label encoding for categorical columns
le = LabelEncoder()
categorical_cols = ['age', 'gender', 'district', 'modeOfTransportation',␣
↪'G03Q13amountOfPeople', 'frequency', 'moneySpent', 'orderingItems',␣
↪'deliveringItems', 'findProducts', 'usingDiscounts', 'preferCash',␣
↪'preferCashless', 'isRelaxing']
for col in categorical_cols:
    df[col] = le.fit_transform(df[col])
```

[83]:
```
# Normalization
# Select the columns for normalization
columns_to_normalize = ['satisGeneralStore', 'satisMusic',␣
↪'satisQualityProducts', 'satisGeneralAssortment']

# Create a scaler object
scaler = StandardScaler()

# Apply normalization to the selected columns
X_normalized = scaler.fit_transform(df[columns_to_normalize])
```

[84]:
```
# Split the dataset into input features (X) and target variable (y)
X = df.drop(columns=['moneySpent'])
y = df['moneySpent']
```

[85]:
```
# Perform data normalization using StandardScaler
scaler = StandardScaler()
X_normalized = scaler.fit_transform(X)
```

```
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_normalized, y,␣
↪test_size=0.2, random_state=42)
```

8 Creating ML model 1

[86]:
```
# Create an instance of the ML model (e.g., Logistic Regression)
model = LogisticRegression()

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

[86]:
```
LogisticRegression()
```

8.1 Prediction on Test data

[87]:
```
# Make predictions on the test set
y_pred = model.predict(X_test)
```

8.2 Model 1 Performance

[88]:
```
# Evaluate the model
# get accuracy of the model
accuracy_score_RegModel = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy_score_RegModel, "percent")
```

```
Accuracy: 0.3181818181818182 percent
```

[89]:
```
#use an othe method
from sklearn.metrics import classification_report, confusion_matrix
confusion_matrix(y_test, model.predict(X_test))
```

[89]:
```
array([[1, 2, 0, 1, 2, 4],
       [0, 9, 0, 6, 0, 5],
       [0, 5, 3, 1, 2, 2],
       [0, 4, 0, 1, 0, 2],
       [0, 3, 0, 2, 3, 0],
       [1, 2, 0, 1, 0, 4]])
```

9 Creating ML model 2

[90]:
```
from sklearn.tree import DecisionTreeClassifier

# Create a Decision Tree classifier
model = DecisionTreeClassifier()
```

[91]:
```
# Train the model
model.fit(X_train, y_train)
```

[91]:
```
DecisionTreeClassifier()
```

9.1 Prediction on Test data

[92]:
```
# Make predictions on the test set
y_pred = model.predict(X_test)
```

9.2 ## Model 2 Performance

[93]:
```
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.3181818181818182
```

[94]:
```
#use an othe method
from sklearn.metrics import classification_report, confusion_matrix
confusion_matrix(y_test, model.predict(X_test))
```

[94]:
```
array([[1, 0, 3, 1, 0, 5],
       [1, 8, 2, 4, 2, 3],
       [2, 3, 6, 1, 0, 1],
       [1, 2, 1, 0, 1, 2],
       [1, 3, 1, 0, 3, 0],
       [1, 0, 1, 0, 3, 3]])
```

10 Report and insight from your analysis

As we se we get low accuracy for both models Logistic regression we acheived 31%,and for Descision tree we get 33%, it seems that Decision tree is model can be used, but we need to try different models until find high accuracy. Even we can try Neural Network models in this data , as we have alot of features and data.

[96]:
```
#print classification report
print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
0	0.14	0.10	0.12	10
1	0.50	0.40	0.44	20
2	0.43	0.46	0.44	13
3	0.00	0.00	0.00	7
4	0.33	0.38	0.35	8
5	0.21	0.38	0.27	8
accuracy			0.32	66
macro avg	0.27	0.29	0.27	66
weighted avg	0.32	0.32	0.32	66