

MileStone_2_Prashant_Raghuwanshi_DSC550

October 16, 2021

Assignment: 7.3 Project Milestone 1 & 2

Name: Prashant Raghuwanshi

Date: 10/16/2021

Course: DSC550-T301 Data Mining (2221-1)

Analyze data to predict the traits to detect Autistics disease among the toddlers Data Source : <https://www.kaggle.com/fabdelja/autism-screening-for-toddlers?select=Toddler+Autism+dataset+July+2018.csv>

Description about Dataset:

The dataset was developed by Dr Fadi Fayeze Thabtah (fadifayeze.com) using a mobile app called ASDTests (ASDtests.com) to screen autism in toddlers. We can use it to estimate the predictive power of machine learning techniques in detecting autistic traits.

Abstract: Autistic Spectrum Disorder (ASD) is a neurodevelopmental condition associated with significant healthcare costs, and early diagnosis can significantly reduce these. Unfortunately, waiting times for an ASD diagnosis are lengthy and procedures are not cost effective. The economic impact of autism and the increase in the number of ASD cases across the world reveals an urgent need for the development of easily implemented and effective screening methods. Therefore, a time-efficient and accessible ASD screening is imminent to help health professionals and inform individuals whether they should pursue formal clinical diagnosis. The rapid growth in the number of ASD cases worldwide necessitates datasets related to behaviour traits.

```
[1]: # Import library
import pandas as pd
import yellowbrick
import matplotlib.pyplot as plt
import numpy as np
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\deprecation.py:144:
FutureWarning: The sklearn.metrics.classification module is deprecated in
version 0.22 and will be removed in version 0.24. The corresponding classes /
functions should instead be imported from sklearn.metrics. Anything that cannot
be imported from sklearn.metrics is now part of the private API.
warnings.warn(message, FutureWarning)
```

Columns Details : Features collected and their descriptions Feature Type Description Variable in Dataset Corresponding Q-chat-10-Toddler Features A1 Does your child look at you when you call his/her name? A2 How easy is it for you to get eye contact with your child? A3 Does your child point to indicate that s/he wants something? (e.g. a toy that is out of reach) A4 Does your child point to share interest with you? (e.g. pointing at an interesting sight) A5 Does your child pretend? (e.g. care for dolls, talk on a toy phone) A6 Does your child follow where you're looking? A7 If you or someone else in the family is visibly upset, does your child show signs of wanting to comfort them? (e.g. stroking hair, hugging them) A8 Would you describe your child's first words as: A9 Does your child use simple gestures? (e.g. wave goodbye) A10 Does your child stare at nothing with no apparent purpose? Age Number Toddlers (months) Score by Q-chat-10 Number 1-10 (Less than or equal 3 no ASD traits; > 3 ASD traits Sex Character Male or Female Ethnicity String List of common ethnicities in text format Born with jaundice Boolean (yes or no) Whether the case was born with jaundice Family member with ASD history Boolean (yes or no) Whether any immediate family member has a PDD Who is completing the test String Parent, self, caregiver, medical staff, clinician ,etc. Why_are_you_taken_the_screening String Use input textbox Class variable String ASD traits or No ASD traits (automatically assigned by the ASDTests app). (Yes / No)

MileStone -1

```
[68]: # 1.Load the data from the "Toddler Autism dataset July 2018.csv" file into a
      ↪ DataFrame.
      addr1 = "D:/MS_DataScience/DSC550/Milestone-1/Toddler Autism dataset July 2018.
      ↪ csv"
      df_todd = pd.read_csv(addr1)
      df_todd.head()
```

```
[68]:
```

	Case_No	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	Age_Mons	Qchat-10-Score	\
0	1	0	0	0	0	0	0	1	1	0	1	28	3	
1	2	1	1	0	0	0	1	1	0	0	0	36	4	
2	3	1	0	0	0	0	0	1	1	0	1	36	4	
3	4	1	1	1	1	1	1	1	1	1	1	24	10	
4	5	1	1	0	1	1	1	1	1	1	1	20	9	

	Sex	Ethnicity	Jaundice	Family_mem_with_ASD	Who completed the test	\
0	f	middle eastern	yes		no	family member
1	m	White European	yes		no	family member
2	m	middle eastern	yes		no	family member
3	m	Hispanic	no		no	family member
4	f	White European	no		yes	family member

	ASD_Traits
0	No
1	Yes
2	Yes
3	Yes
4	Yes

```
[4]: df_todd.info() # find out the datatype for each columns
```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1054 entries, 0 to 1053
Data columns (total 19 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   Case_No                                1054 non-null   int64
1   A1                                     1054 non-null   int64
2   A2                                     1054 non-null   int64
3   A3                                     1054 non-null   int64
4   A4                                     1054 non-null   int64
5   A5                                     1054 non-null   int64
6   A6                                     1054 non-null   int64
7   A7                                     1054 non-null   int64
8   A8                                     1054 non-null   int64
9   A9                                     1054 non-null   int64
10  A10                                    1054 non-null   int64
11  Age_Mons                              1054 non-null   int64
12  Qchat-10-Score                        1054 non-null   int64
13  Sex                                    1054 non-null   object
14  Ethnicity                             1054 non-null   object
15  Jaundice                              1054 non-null   object
16  Family_mem_with_ASD                   1054 non-null   object
17  Who completed the test                 1054 non-null   object
18  ASD_Traits                            1054 non-null   object
dtypes: int64(13), object(6)
memory usage: 156.6+ KB

```

```

[5]: #5.      Look at summary information about your data (total, mean, min, max,
      ↪freq, unique, etc.) Does this present any more questions for you? Does it
      ↪lead you to a conclusion yet?
print("\nDescribe Data\n")
print(df_todd.describe())

```

Describe Data

	Case_No	A1	A2	A3	A4 \
count	1054.000000	1054.000000	1054.000000	1054.000000	1054.000000
mean	527.500000	0.563567	0.448767	0.401328	0.512334
std	304.407895	0.496178	0.497604	0.490400	0.500085
min	1.000000	0.000000	0.000000	0.000000	0.000000
25%	264.250000	0.000000	0.000000	0.000000	0.000000
50%	527.500000	1.000000	0.000000	0.000000	1.000000
75%	790.750000	1.000000	1.000000	1.000000	1.000000
max	1054.000000	1.000000	1.000000	1.000000	1.000000

	A5	A6	A7	A8	A9 \
count	1054.000000	1054.000000	1054.000000	1054.000000	1054.000000

mean	0.524668	0.576850	0.649905	0.459203	0.489564
std	0.499628	0.494293	0.477226	0.498569	0.500128
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000
50%	1.000000	1.000000	1.000000	0.000000	0.000000
75%	1.000000	1.000000	1.000000	1.000000	1.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000

	A10	Age_Mons	Qchat-10-Score
count	1054.000000	1054.000000	1054.000000
mean	0.586338	27.867173	5.212524
std	0.492723	7.980354	2.907304
min	0.000000	12.000000	0.000000
25%	0.000000	23.000000	3.000000
50%	1.000000	30.000000	5.000000
75%	1.000000	36.000000	8.000000
max	1.000000	36.000000	10.000000

```
[6]: print("\nSummarized Data\n")
      print(df_todd.describe(include=['O']))
```

Summarized Data

	Sex	Ethnicity	Jaundice	Family_mem_with_ASF \
count	1054	1054	1054	1054
unique	2	11	2	2
top	m	White European	no	no
freq	735	334	766	884

	Who completed the test	ASD_Traits
count	1054	1054
unique	5	2
top	family member	Yes
freq	1018	728

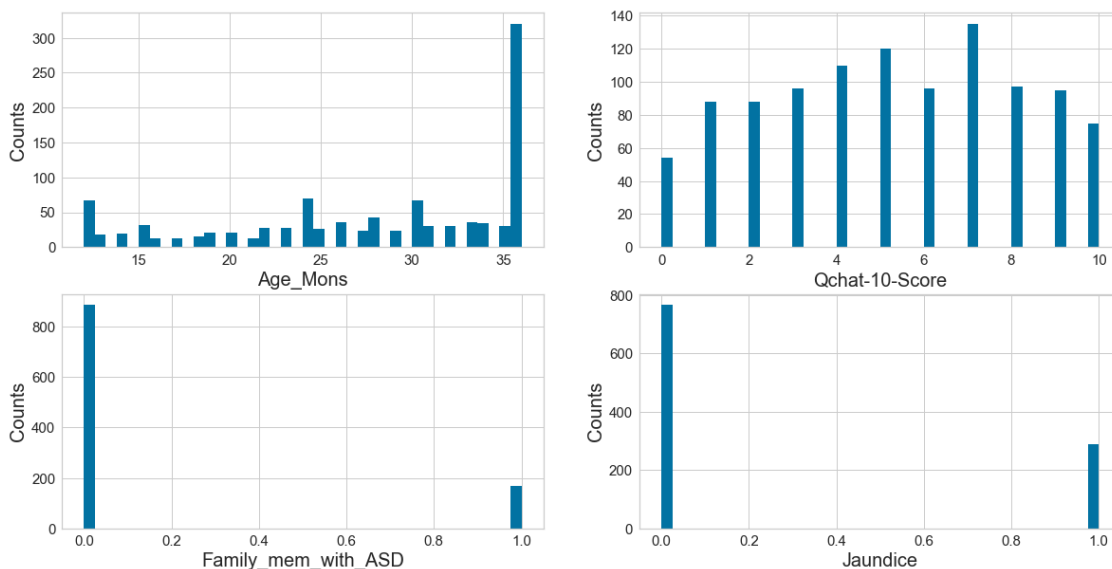
```
[7]: # converting categorical data to numbers
df_todd['ASD_Traits'] = df_todd['ASD_Traits'].replace(['Yes', 'No'], [1, 0])
df_todd['Jaundice'] = df_todd['Jaundice'].replace(['yes', 'no'], [1, 0])
df_todd['Family_mem_with_ASF'] = df_todd['Family_mem_with_ASF'].replace(['yes', 'no'], [1, 0])
```

```
[8]: #6. Make some histograms of your data ("A picture is worth a thousand
      words!")
# Specify the features of interest
num_features = ['Age_Mons', 'Qchat-10-Score', 'Family_mem_with_ASF', 'Jaundice']
xaxes = num_features
yaxes = ['Counts', 'Counts', 'Counts', 'Counts']
```

```

# set up the figure size
plt.rcParams['figure.figsize'] = (20, 10)
# make subplots
fig, axes = plt.subplots(nrows = 2, ncols = 2)
# draw histograms
axes = axes.ravel()
for idx, ax in enumerate(axes):
    ax.hist(df_todd[num_features[idx]].dropna(), bins=40)
    ax.set_xlabel(xaxes[idx], fontsize=20)
    ax.set_ylabel(yaxes[idx], fontsize=20)
    ax.tick_params(axis='both', labelsize=15)
plt.show()

```



```

[9]: #7: Barcharts: set up the figure size
%matplotlib inline
plt.rcParams['figure.figsize'] = (20, 10)

# make subplots
fig, axes = plt.subplots(nrows = 2, ncols = 2)

# make the data read to feed into the visulizer
X_ASD_Traits = df_todd.groupby('ASD_Traits').size().
    ↳reset_index(name='Counts')['ASD_Traits']
Y_ASD_Traits = df_todd.groupby('ASD_Traits').size().
    ↳reset_index(name='Counts')['Counts']

# make the bar plot
axes[0, 0].bar(X_ASD_Traits, Y_ASD_Traits)
axes[0, 0].set_title('ASD_Traits', fontsize=25)

```

```

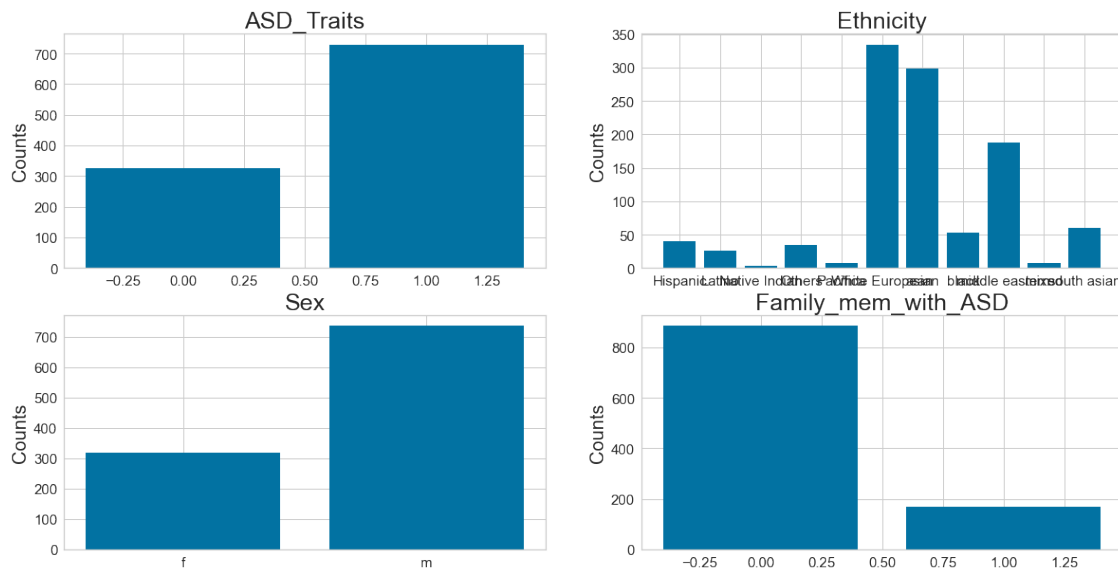
axes[0, 0].set_ylabel('Counts', fontsize=20)
axes[0, 0].tick_params(axis='both', labelsize=15)

# make the data read to feed into the visulizer
X_Ethnicity = df_todd.groupby('Ethnicity').size().
    ↳reset_index(name='Counts')['Ethnicity']
Y_Ethnicity = df_todd.groupby('Ethnicity').size().
    ↳reset_index(name='Counts')['Counts']
# make the bar plot
axes[0, 1].bar(X_Ethnicity, Y_Ethnicity)
axes[0, 1].set_title('Ethnicity', fontsize=25)
axes[0, 1].set_ylabel('Counts', fontsize=20)
axes[0, 1].tick_params(axis='both', labelsize=15)

# make the data read to feed into the visualizer
X_Sex = df_todd.groupby('Sex').size().reset_index(name='Counts')['Sex']
Y_Sex = df_todd.groupby('Sex').size().reset_index(name='Counts')['Counts']
# make the bar plot
axes[1, 0].bar(X_Sex, Y_Sex)
axes[1, 0].set_title('Sex', fontsize=25)
axes[1, 0].set_ylabel('Counts', fontsize=20)
axes[1, 0].tick_params(axis='both', labelsize=15)

# make the data read to feed into the visualizer
X_Family_mem_with_ASD = df_todd.groupby('Family_mem_with_ASD').size().
    ↳reset_index(name='Counts')['Family_mem_with_ASD']
Y_Family_mem_with_ASD = df_todd.groupby('Family_mem_with_ASD').size().
    ↳reset_index(name='Counts')['Counts']
# make the bar plot
axes[1, 1].bar(X_Family_mem_with_ASD, Y_Family_mem_with_ASD)
axes[1, 1].set_title('Family_mem_with_ASD', fontsize=25)
axes[1, 1].set_ylabel('Counts', fontsize=20)
axes[1, 1].tick_params(axis='both', labelsize=15)
plt.show()

```

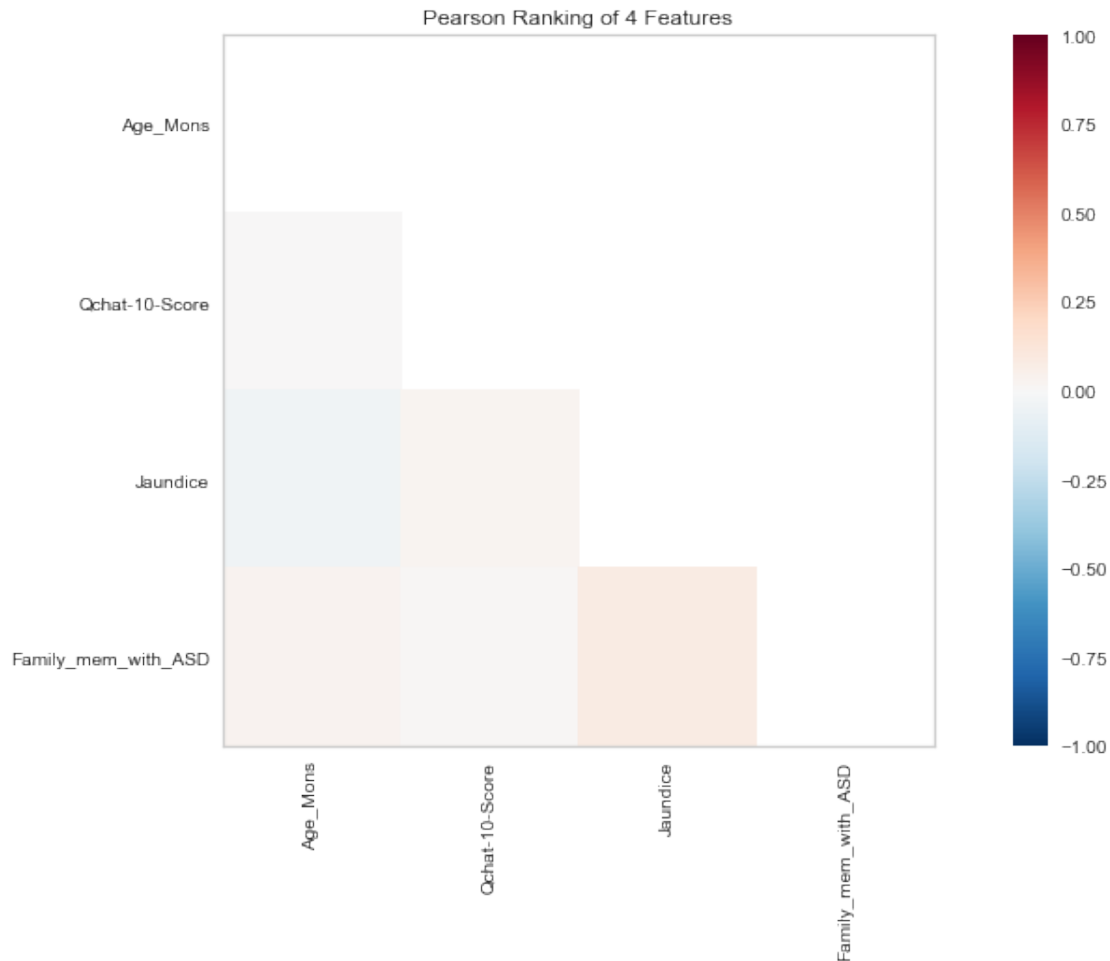


```
[19]: # The correlation between the variables is low (1 or -1 is high positive or
      ↪ high negative, 0 is low or no correlation)
      # These results show there is positive correlation between 'ASD_Traits' &
      ↪ 'Qchat-10-Score', but it's not a high correlation among other variables.

      #Step 8: Pearson Ranking
      #set up the figure size
      #matplotlib inline
      plt.rcParams['figure.figsize'] = (15, 7)

      # import the package for visulization of the correlation
      from yellowbrick.features import Rank2D
      num_features = ['Age_Mons', 'Qchat-10-Score', 'Jaundice', 'Family_mem_with_ASD']
      # Define features to test for correlation
      # extract the numpy arrays from the data frame
      X = df_todd[num_features].to_numpy()

      # instantiate the visualizer with the Covariance ranking algorithm
      visualizer = Rank2D(features=num_features, algorithm='pearson')
      visualizer.fit(X) # Fit the data to the visualizer
      visualizer.transform(X) # Transform the data
      visualizer.poof(outpath="pcoords1.png") # Draw/show/poof the data
      plt.show()
```



```
[28]: # Use Parallel Coordinates visualization to compare the distributions of
      ↪ numerical variables between
      # In a Parallel Coordinates Plot, each variable is given its own axis and all
      ↪ the axes are placed in parallel to each other.
      # Values are plotted as a series of lines that connected across all the axes.
      ↪ This means that each line is a collection of points placed on each axis,
      ↪ that have all been connected together.
      # Parallel Coordinates Plots are ideal for comparing many variables together
      ↪ and seeing the relationships between them.
      plt.rcParams['figure.figsize'] = (15, 7)
      plt.rcParams['font.size'] = 50
      # setup the color for yellowbrick vizualizer
      from yellowbrick.style import set_palette
      set_palette('sns_bright')

      # import packages
```



```

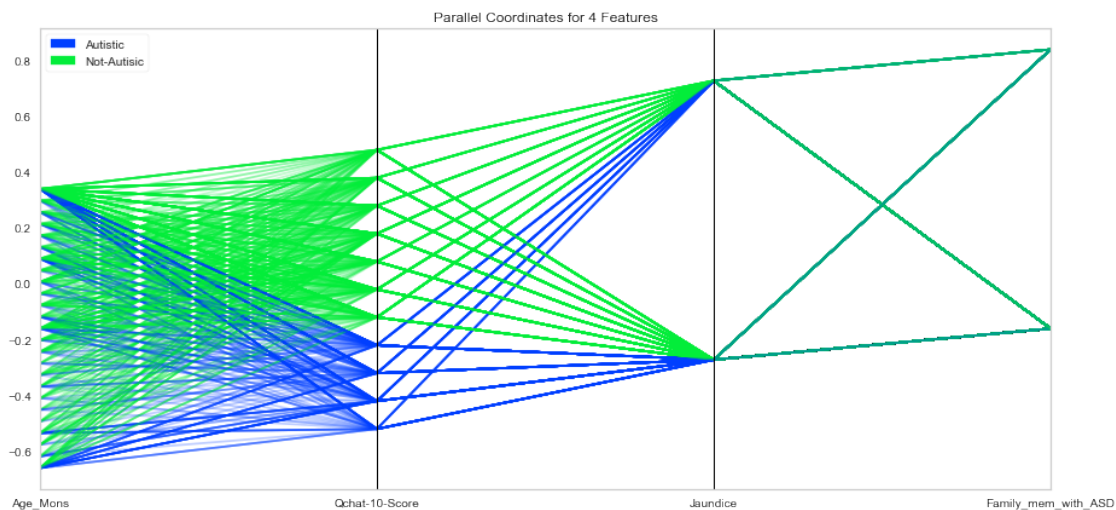
from yellowbrick.features import ParallelCoordinates
# Specify the features of interest and the classes of the target
classes = ['Autistic', 'Not-Autistic']
num_features = ['Age_Mons', 'Qchat-10-Score', 'Jaundice', 'Family_mem_with_ASF']
# copy data to a new dataframe
data_norm = df_todd.copy()
# Each axis can have a different scale, as each variable works off a different
    ↳ unit of measurement,
# or all the axes can be normalised to keep all the scales uniform..
# normalize data to 0-1 range
for feature in num_features:
    data_norm[feature] = (df_todd[feature] - df_todd[feature].
    ↳ mean(skipna=True)) / (df_todd[feature].max(skipna=True) - df_todd[feature].
    ↳ min(skipna=True))

# Extract the numpy arrays from the data frame
X = data_norm[num_features].to_numpy()
y = df_todd.ASD_Traits.to_numpy()

# Instantiate the visualizer

visualizer = ParallelCoordinates(classes=classes, features=num_features)
visualizer.fit(X, y)      # Fit the data to the visualizer
visualizer.transform(X)   # Transform the data
visualizer.poof(outpath="d://pcoords2.png") # Draw/show/poof the data
plt.show();

```



Parallel coordinate for 4 features shows below information

as per the graph fro autistic patients,we have a seen relationship between age ,score and jaundice variables,how ever family men with ads feature is not having any relation with other features

for non austics, we have sees relationship beteen all listed 4 features

```
[13]: # Use Stack Bar Charts to compare toddlers who is having ASD & who didn't have
      ↳ ASD based on the other variables.

      # Step 10 - stacked bar charts to compare autistic/not autistic
      #set up the figure size
      #%matplotlib inline
      plt.rcParams['figure.figsize'] = (20, 10)

      # make subplots
      fig, axes = plt.subplots(nrows = 2, ncols = 2)

      # make the data read to feed into the visulizer
      Sex_autism = df_todd[df_todd['ASD_Traits']==1]['Sex'].value_counts()
      Sex_not_autism = df_todd[df_todd['ASD_Traits']==0]['Sex'].value_counts()
      Sex_not_autism = Sex_not_autism.reindex(index = Sex_autism.index)
      # make the bar plot
      p1 = axes[0, 0].bar(Sex_autism.index, Sex_autism.values)
      p2 = axes[0, 0].bar(Sex_not_autism, Sex_not_autism.values, bottom=Sex_autism.
      ↳values)
      axes[0, 0].set_title('Sex', fontsize=25)
      axes[0, 0].set_ylabel('Counts', fontsize=20)
      axes[0, 0].tick_params(axis='both', labels=15)
      axes[0, 0].legend((p1[0], p2[0]), ('Autistic', 'Not-Autistic'), fontsize = 15)

      # make the data read to feed into the visualizer
      ethnicity_autism = df_todd[df_todd['ASD_Traits']==1]['Ethnicity'].value_counts()
      ethnicity_not_autism = df_todd[df_todd['ASD_Traits']==0]['Ethnicity'].
      ↳value_counts()
      ethnicity_not_autism = ethnicity_not_autism.reindex(index = ethnicity_autism.
      ↳index)
      # make the bar plot
      p3 = axes[0, 1].bar(ethnicity_autism.index, ethnicity_autism.values)
      p4 = axes[0, 1].bar(ethnicity_not_autism.index, ethnicity_not_autism.values,
      ↳bottom=ethnicity_autism.values)
      axes[0, 1].set_title('Ethnicity', fontsize=25)
      axes[0, 1].set_ylabel('Counts', fontsize=20)
      axes[0, 1].tick_params(axis='both', labels=15)
      axes[0, 1].legend((p3[0], p4[0]), ('Autistic', 'Not-Autistic'), fontsize = 15)

      # make the data read to feed into the visualizer
      ASD_autism = df_todd[df_todd['ASD_Traits']==1]['Family_mem_with_ASD'].
      ↳value_counts()
```

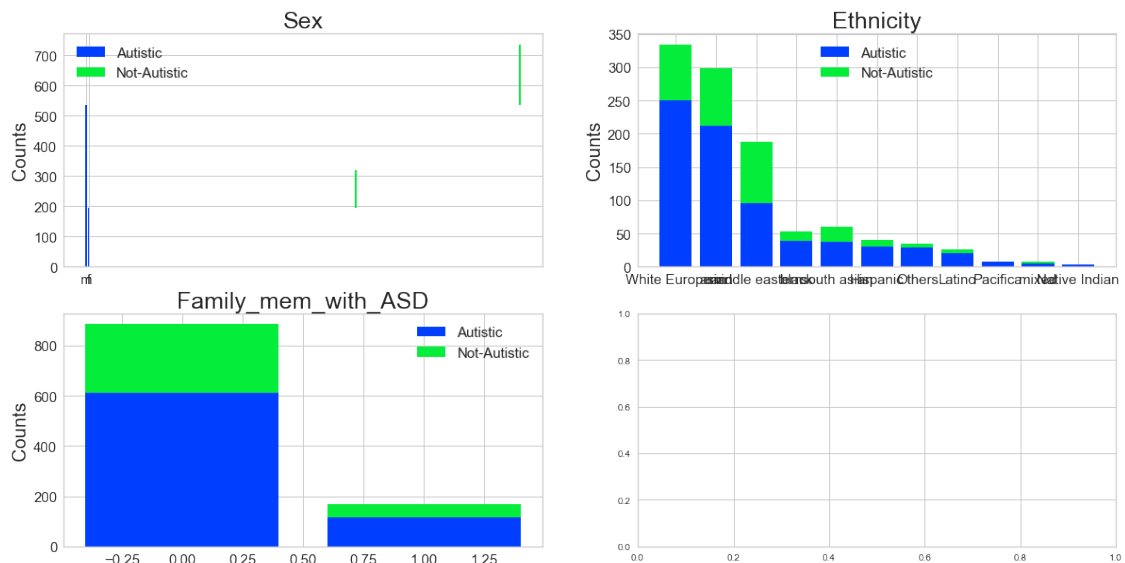
```

ASD_not_autism = df_todd[df_todd['ASD_Traits']==0]['Family_mem_with_ASD'].
    ↳value_counts()
ASD_not_autism = ASD_not_autism.reindex(index = ASD_autism.index)

# make the bar plot
p5 = axes[1, 0].bar(ASD_autism.index, ASD_autism.values)
p6 = axes[1, 0].bar(ASD_not_autism.index, ASD_not_autism.values,
    ↳bottom=ASD_autism.values)
axes[1, 0].set_title('Family_mem_with_ASD', fontsize=25)
axes[1, 0].set_ylabel('Counts', fontsize=20)
axes[1, 0].tick_params(axis='both', labelsize=15)
axes[1, 0].legend((p5[0], p6[0]), ('Autistic', 'Not-Autistic'), fontsize = 15)

```

[13]: <matplotlib.legend.Legend at 0x1f182ca4eb0>



less females have ASD as compared to MEN, white european is having more rate for insfaction from ASD,

Family with ASD history and with Non ASD history, in both toddlers have ASD

Milestone 2

1) drop any features that are not useful for your model building. You should explain and justify why the feature dropped is not useful

2) address any missing data issues.

3) Build any new features that you need for your model, e.g., create dummy variables for categorical features if necessary. Explain your process at each step. You can use any methods/tools you think are most appropriate.

```
[31]: #fill the missing data with median value
# num_features = ['Age_Mons', 'Qchat-10-Score', 'Jaundice',
↳ 'Family_mem_with_ASD']
def fill_na_median(df_todd, inplace=True):
    return df_todd.fillna(df_todd.median(), inplace=inplace)

fill_na_median(df_todd['Qchat-10-Score'])

# check the result
print(df_todd['Qchat-10-Score'].describe())
```

```
count      1054.000000
mean         5.212524
std          2.907304
min           0.000000
25%           3.000000
50%           5.000000
75%           8.000000
max          10.000000
Name: Qchat-10-Score, dtype: float64
```

```
[60]: # fill with the most represented value
def fill_na_most(df_todd, inplace=True):
    return df_todd.fillna('family member', inplace=inplace)

fill_na_most(df_todd['Who completed the test'])

# check the result
print(df_todd['Who completed the test'].describe())
```

```
count           1054
unique             5
top      family member
freq             1018
Name: Who completed the test, dtype: object
```

```
[69]: np.unique(df_todd['Who completed the test'])
```

```
[69]: array(['Health Care Professional', 'Health care professional', 'Others',
        'Self', 'family member'], dtype=object)
```

found data quality issue in one of the variable value of df_todd['Who completed the test'] . so replace the values

```
[70]: df_todd['Who completed the test'] = df_todd['Who completed the test'].
↳ replace(['Health Care Professional'], 'Health care professional')
```

```
[71]: np.unique(df_todd['Who completed the test'])
```

```
[71]: array(['Health care professional', 'Others', 'Self', 'family member'],
      dtype=object)
```

```
[72]: # check the result
print(df_todd['Who completed the test'].describe())
```

```
count          1054
unique           4
top    family member
freq          1018
Name: Who completed the test, dtype: object
```

```
[73]: # fill with the most represented value
def fill_na_most(df_todd, inplace=True):
    return df_todd.fillna('White European', inplace=inplace)

fill_na_most(df_todd['Ethnicity'])

# check the result
print(df_todd['Ethnicity'].describe())
```

```
count          1054
unique          11
top    White European
freq           334
Name: Ethnicity, dtype: object
```

```
[41]: # import package
import numpy as np

# log-transformation
def log_transformation(df_todd):
    return df_todd.apply(np.log1p)

df_todd['Qchat-10-Score_log'] = log_transformation(df_todd['Qchat-10-Score'])

# check the data
print(df_todd.describe())
```

	Case_No	A1	A2	A3	A4 \
count	1054.000000	1054.000000	1054.000000	1054.000000	1054.000000
mean	527.500000	0.563567	0.448767	0.401328	0.512334
std	304.407895	0.496178	0.497604	0.490400	0.500085
min	1.000000	0.000000	0.000000	0.000000	0.000000
25%	264.250000	0.000000	0.000000	0.000000	0.000000
50%	527.500000	1.000000	0.000000	0.000000	1.000000
75%	790.750000	1.000000	1.000000	1.000000	1.000000

max	1054.000000	1.000000	1.000000	1.000000	1.000000
-----	-------------	----------	----------	----------	----------

	A5	A6	A7	A8	A9 \
count	1054.000000	1054.000000	1054.000000	1054.000000	1054.000000
mean	0.524668	0.576850	0.649905	0.459203	0.489564
std	0.499628	0.494293	0.477226	0.498569	0.500128
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000
50%	1.000000	1.000000	1.000000	0.000000	0.000000
75%	1.000000	1.000000	1.000000	1.000000	1.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000

	A10	Age_Mons	Qchat-10-Score	Jaundice \
count	1054.000000	1054.000000	1054.000000	1054.000000
mean	0.586338	27.867173	5.212524	0.273245
std	0.492723	7.980354	2.907304	0.445837
min	0.000000	12.000000	0.000000	0.000000
25%	0.000000	23.000000	3.000000	0.000000
50%	1.000000	30.000000	5.000000	0.000000
75%	1.000000	36.000000	8.000000	1.000000
max	1.000000	36.000000	10.000000	1.000000

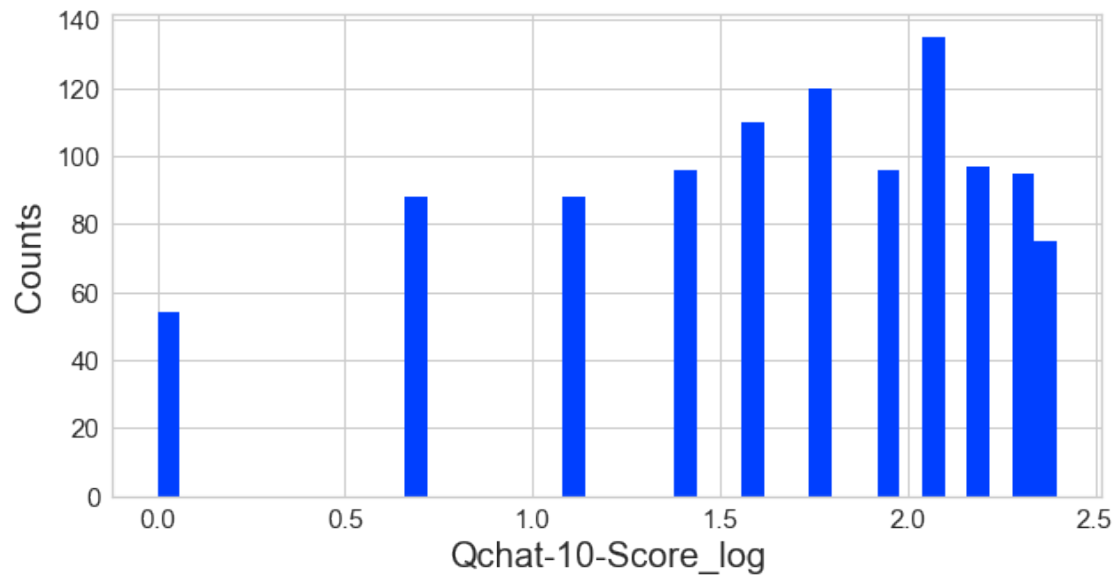
	Family_mem_with_ASD	ASD_Traits	Qchat-10-Score_log
count	1054.000000	1054.000000	1054.000000
mean	0.161290	0.690702	1.671783
std	0.367973	0.462424	0.622364
min	0.000000	0.000000	0.000000
25%	0.000000	0.000000	1.386294
50%	0.000000	1.000000	1.791759
75%	0.000000	1.000000	2.197225
max	1.000000	1.000000	2.397895

above results shows the new feature qchart-1-score_log columns which contains Log Transformed values for highly skewed data

```
[42]: # Log Transformation is a good method to use on highly skewed data.
#check the distribution using histogram
# set up the figure size
#adjust skewed data (result)

plt.rcParams['figure.figsize'] = (10, 5)

plt.hist(df_todd['Qchat-10-Score_log'], bins=40)
plt.xlabel('Qchat-10-Score_log', fontsize=20)
plt.ylabel('Counts', fontsize=20)
plt.tick_params(axis='both', labelsize=15)
plt.show()
```



log transformed high skewed values & its counts are showing in histogram

```
[74]: # convert categorical data to numbers
      #get the categorical data
      cat_features = ['Who completed the test', 'Sex']
      data_cat = df_todd[cat_features]
      # One Hot Encoding
      data_cat_dummies = pd.get_dummies(data_cat)
      # check the data
      print(data_cat_dummies.head(8))
```

```
Who completed the test_Health care professional \
0 0
1 0
2 0
3 0
4 0
5 0
6 0
7 0

Who completed the test_Others Who completed the test_Self \
0 0 0
1 0 0
2 0 0
3 0 0
4 0 0
5 0 0
6 0 0
```

```
7                                0                                0
```

```
Who completed the test_family member  Sex_f  Sex_m
0                                1      1      0
1                                1      0      1
2                                1      0      1
3                                1      0      1
4                                1      1      0
5                                1      0      1
6                                1      0      1
7                                1      0      1
```

created new dummy variables , Who completed the test_Health Care Professional,
Who completed the test_Others, Who completed the test_Self, Who completed the
test_family member,Sex_f,Sex_m

```
[ ]:
```