Project 5

Alice Pao

Project Summary

In this project, I cleaned the star wars survey data by changing column names, replacing missing data and recreate charts to confirm the accuracy of the imported data. Moreover, I used the wrangled data as foundation to build a machine learning model to predict whether a person's income is more than 50000.

Technical Details

Grand Question 1

Shorten the column names and clean them up for easier use with pandas.

I renamed the column names by dividing dataset into 2 pieces. One with the first two rows(questions and options) and the other one is answer rows.

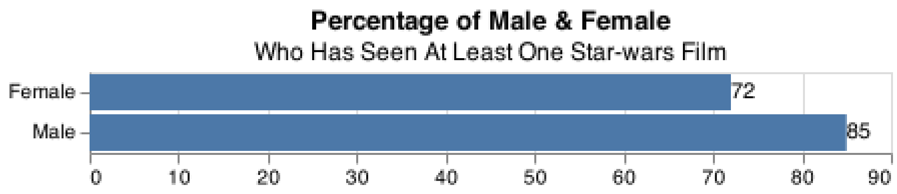
I used replace function to simplify the questions and options. Here are the final column names:

```
Index(['id', 'seen_any', 'is_fan', 'seen_I_the_phantom_menace',
       'seen_II_attack_of_the_clones', 'seen_III_revenge_of_the_sith',
       'seen_IV_a_new_hope', 'seen_V_the_empire_strikes_back',
       'seen_VI_return_of_the_jedi', 'rank_I_the_phantom_menace',
       'rank_II_attack_of_the_clones', 'rank_III_revenge_of_the_sith',
       'rank_IV_a_new_hope', 'rank_V_the_empire_strikes_back',
       'rank_VI_return_of_the_jedi', 'rate_han', 'rate_luke', 'rate_leia',
       'rate_skywalker', 'rate_obi', 'rate_palpatine', 'rate_vader',
       'rate_lando', 'rate_fett', 'rate_c3po', 'rate_r2d2', 'rate_jar',
       'rate_amidala', 'rate_yoda', 'first_shot_character', 'know_universe',
       'is_universe_fan  ', 'is_franchise_fan', 'gender', 'age', 'income',
       'education', 'region'],
      dtype='object')
```

Grand Question 2

Please validate that the data provided on GitHub lines up with the article by recreating 2 of their visuals and calculating 2 summaries that they report in the article.

Summary 1: 85 percent of men have seen at least one "Star Wars" film compared to 72 percent of women



Summary 2: 72 percent of men compared to 60 percent of women consider themselves Star Wars fans out of those who have seen at least one film

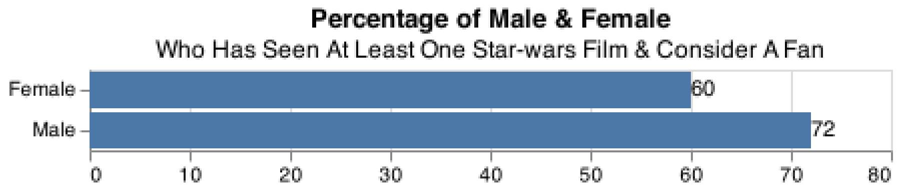


Chart 1:

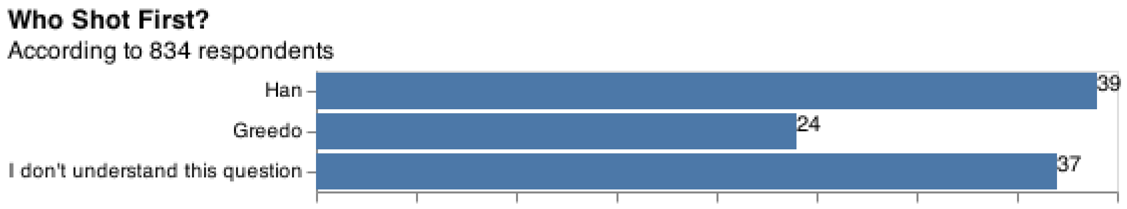
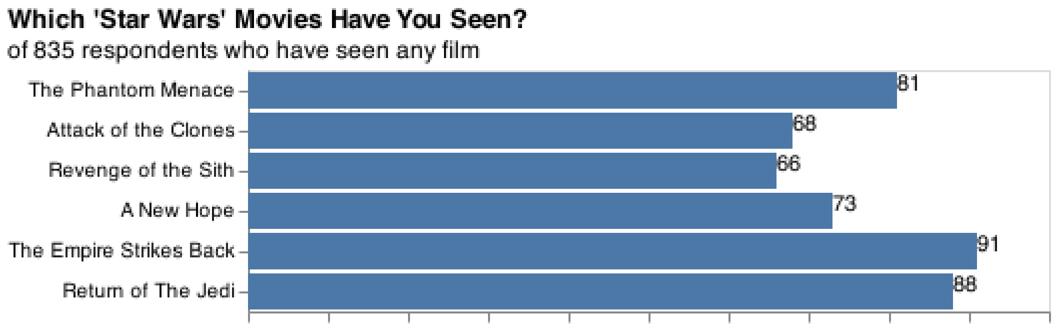


Chart 2:

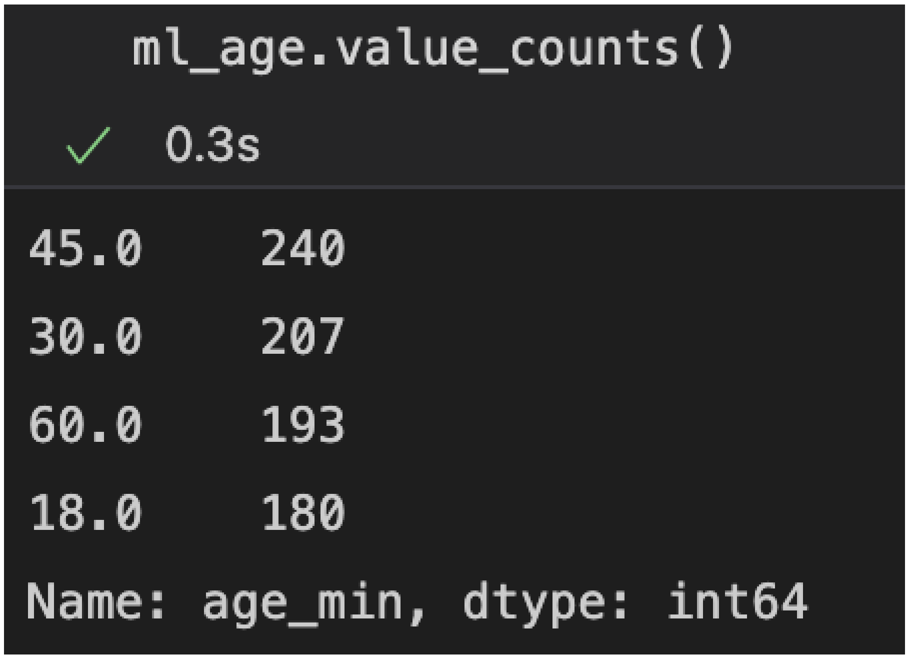


Grand Question 3

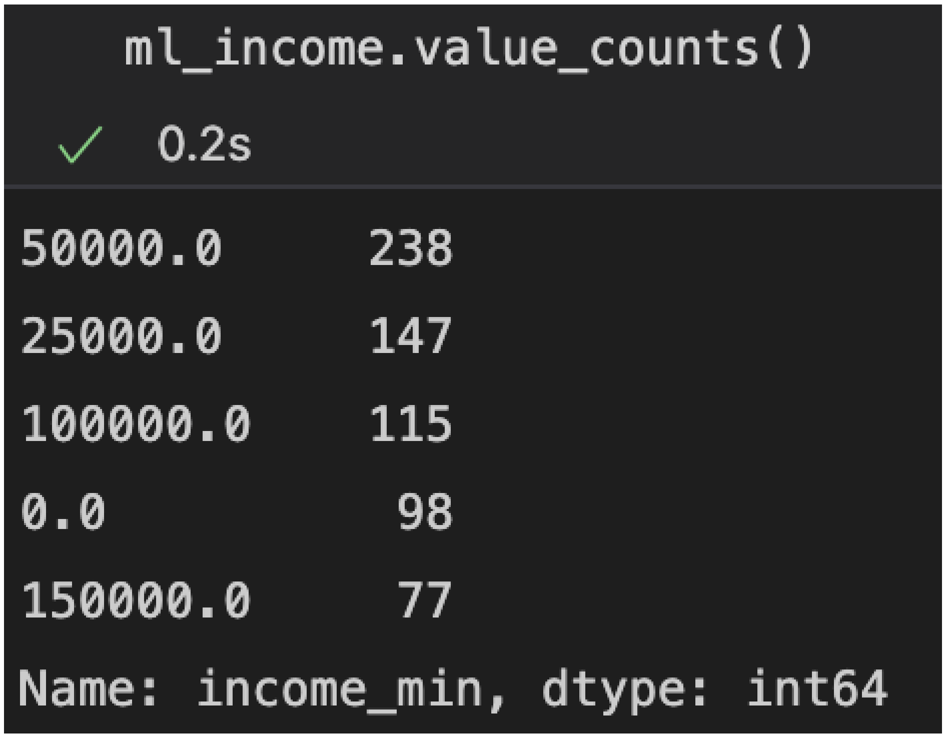
Clean and format the data so that it can be used in a machine learning model. As you format the data, you should complete each item listed below. In your final report provide example(s) of the reformatted data with a short description of the changes made.

I first filter the dataset to those who have seen at least 1 film. Next, I created new columns for age, income, and education, changing ranges to single numbers for each 3 columns. To be more specific, age column now only has thse four categories: 18, 30, 60, 45; income has 0, 25000, 50000, 100000, 150000; education has 10, 12, 14, 16, 20 category.

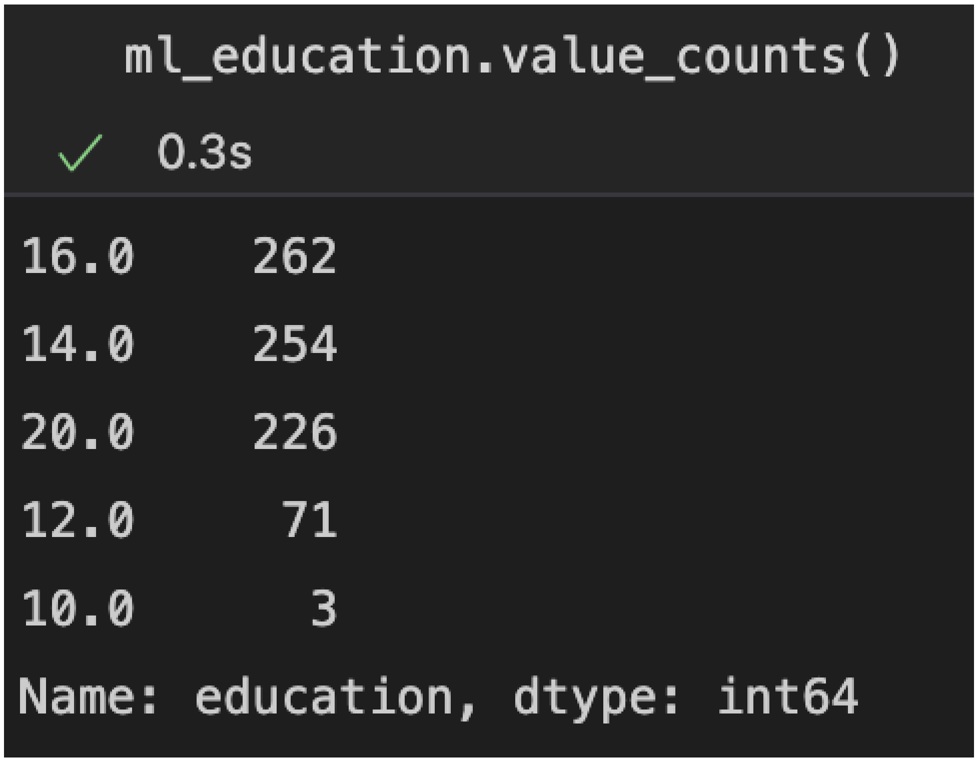
Lastly, change the rest of columns into machine learning language, 0 and 1.



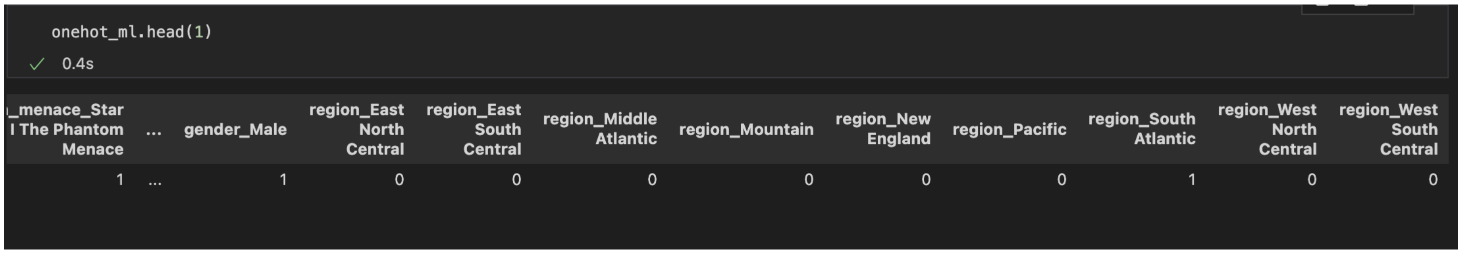
```
ml_age.value_counts()
✓ 0.3s
45.0    240
30.0    207
60.0    193
18.0    180
Name: age_min, dtype: int64
```



```
ml_income.value_counts()
✓ 0.2s
50000.0     238
25000.0     147
100000.0    115
0.0          98
150000.0     77
Name: income_min, dtype: int64
```



```
ml_education.value_counts()
✓ 0.3s
16.0    262
14.0    254
20.0    226
12.0     71
10.0      3
Name: education, dtype: int64
```



```
onehot_ml.head(1)
✓ 0.4s
```

_menace_Star I The Phantom Menace	...	gender_Male	region_East North Central	region_East South Central	region_Middle Atlantic	region_Mountain	region_New England	region_Pacific	region_South Atlantic	region_West North Central	region_West South Central
1	...	1	0	0	0	0	0	0	1	0	0

Grand Question 4

Build a machine learning model that predicts whether a person makes more than 50k. Describe your model and report the accuracy.

Before buiding the machine learning model, I created target(y), income > 5000 and dropped income column for features (x).

I chose RandomForestClassifier as classifier to train my data. The final accuracy I got is 70%.

Appendix A

```
#%%
# import packages
import pandas as pd
import altair as alt
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics

#%%
# import csv data
df = pd.read_csv('https://github.com/fivethirtyeight/data/raw/master/star-wars-survey/StarWars.csv', en
#%%
# Grand Question 1: clean the data
# get the rest starting with 3rd row
df_rows = pd.read_csv('https://github.com/fivethirtyeight/data/raw/master/star-wars-survey/StarWars.csv

# get the first 2 rows
df_title = pd.read_csv('https://github.com/fivethirtyeight/data/raw/master/star-wars-survey/StarWars.cs
# %%
# rename column names
question = df_title.iloc[0,:]
question = (question
                # .str.replace treats symbols (regular expression) differently. eg, ?, +
                .str.replace('RespondentID', 'id')
                .replace('Have you seen any of the 6 films in the Star Wars franchise?', 'seen_any')
                .replace('Do you consider yourself to be a fan of the Star Wars film franchise?', 'is_f
                .replace('Which of the following Star Wars films have you seen? Please select all that
                .fillna(method = 'ffill')
                .replace('Please rank the Star Wars films in order of preference with 1 being your favo
                .fillna(method = 'ffill')
                .replace('Please state whether you view the following characters favorably, unfavorably
                .fillna(method = 'ffill')
                .replace('Which character shot first?', 'first_shot_character')
                .replace('Are you familiar with the Expanded Universe?', 'know_universe')
                .str.replace('Do you consider yourself to be a fan of the Expanded Universe\?', 'is_uni
                .str.replace('æ', '')
                .replace('Do you consider yourself to be a fan of the Star Trek franchise?', 'is_franch
                .replace('Gender', 'gender')
                .replace('Age', 'age')
                .replace('Household Income', 'income')
                .replace('Education', 'education')
                .replace('Location (Census Region)', 'region')
                #.ffill
            )
question

# %%
# pull up a row of data [row, column]
# df_title.iloc[0,:]
# clean the questions
response = df_title.iloc[1,:]
response = (response
                # .str.replace treats symbols (regular expression) differently. eg, ?, +
                # replace only does the exact characters
                .str.replace('Response', '')
                .str.replace('Star Wards: Episode ', 'I_the_phantom_menace')
                .replace('Star Wars: Episode II  Attack of the Clones', 'II_attack_of_the_clones')
                .replace('Star Wars: Episode III  Revenge of the Sith', 'III_revenge_of_the_sith')
                .replace('Star Wars: Episode IV  A New Hope', 'IV_a_new_hope')
                .replace('Star Wars: Episode V The Empire Strikes Back', 'V_the_empire_strikes_back')
                .replace('Star Wars: Episode VI Return of the Jedi', 'VI_return_of_the_jedi')
                .replace('Star Wars: Episode I  The Phantom Menace', 'I_the_phantom_menace')
                .replace('Star Wars: Episode II  Attack of the Clones', 'II_attack_of_the_clones')
                .replace('Star Wars: Episode III  Revenge of the Sith', 'III_revenge_of_the_sith')
                .replace('Star Wars: Episode IV  A New Hope', 'IV_a_new_hope')
                .replace('Star Wars: Episode V The Empire Strikes Back', 'V_the_empire_strikes_back')
                .replace('Star Wars: Episode VI Return of the Jedi', 'VI_return_of_the_jedi')
                .replace('Han Solo', 'han')
                .replace('Luke Skywalker', 'luke')
                .replace('Princess Leia Organa', 'leia')
                .replace('Anakin Skywalker', 'skywalker')
                .replace('Obi Wan Kenobi', 'obi')
                .replace('Emperor Palpatine', 'palpatine')
                .replace('Darth Vader','vader')
                .replace('Lando Calrissian', 'lando')
                .replace('Boba Fett', 'fett')
                .replace('C-3P0', 'c3po')
                .replace('R2 D2', 'r2d2')
                .replace('Jar Jar Binks', 'jar')
                .replace('Padme Amidala', 'amidala')
                .replace('Yoda', 'yoda')
                .replace(np.nan, '')
            )
response

# %%
# add df_title 2 rows together
column_names = question + response
column_names

# %%
# comobine
df_rows.columns = column_names
df_rows_columns_table = df_rows.columns

normalize=True

#%%
# Grand Question 2: summary1, male vs female seen_any
# remember to specify what column of vlaues you want after groupby()
```

```
seen_any_gender = df_rows.filter(['seen_any', 'gender']).dropna().groupby('gender').seen_any.value_coun
seen_any_gender_df = seen_any_gender.reset_index(name='value')
seen_any_gender_yes = seen_any_gender_df.query('seen_any == "Yes"').reset_index()
# create chart
q2summarychart1 = (alt.Chart(seen_any_gender_yes)
                        .mark_bar()
                        .encode(
                            x = alt.X('value', axis=alt.Axis(title='')),
                            y = alt.Y('gender', axis=alt.Axis(title='')))
)
q2summarychart1_1 = (alt.Chart(seen_any_gender_yes)
                        .mark_text(align='left', baseline='middle', dx=0, dy=0)
                        .encode(
                            text= alt.Text('value:Q'),
                            x = alt.X('value', axis=alt.Axis(title='')),
                            y = alt.Y('gender', axis=alt.Axis(title='')))
)
q2summarychart1f = (q2summarychart1 + q2summarychart1_1).properties(title={
                    "text": "Percentage of Male & Female",
                    "subtitle": "Who Has Seen At Least One Star-wars Film",
                    })
q2summarychart1f.save('question_2_1.png')
# Grand Question 2: summary2, male vs femal seen_any >> is_fan
seen_any_fan = df_rows.filter(['seen_any', 'gender', 'is_fan']).dropna().groupby('gender').is_fan.value
seen_any_fan_df = seen_any_fan.reset_index(name='value')
seen_any_fan_yes = seen_any_fan_df.query('is_fan == "Yes"').reset_index()
# create chart
q2summarychart2 = (alt.Chart(seen_any_fan_yes)
                        .mark_bar()
                        .encode(
                            x = alt.X('value', axis=alt.Axis(title='')),
                            y = alt.Y('gender', axis=alt.Axis(title='')))
)
q2summarychart2_1 = (alt.Chart(seen_any_fan_yes)
                        .mark_text(align='left', baseline='middle', dx=0, dy=0)
                        .encode(
                            text= alt.Text('value:Q'),
                            x = alt.X('value', axis=alt.Axis(title='')),
                            y = alt.Y('gender', axis=alt.Axis(title='')))
)
q2summarychart2f = (q2summarychart2 + q2summarychart2_1).properties(title={
                    "text": "Percentage of Male & Female",
                    "subtitle": "Who Has Seen At Least One Star-wars Film & Consider A Fan",
                    })
q2summarychart2f.save('question_2_2.png')

# Grand Question 2: chart for which character got shot first
first_shot = df_rows.value_counts(['first_shot_character'], normalize=True).round(2) * 100
first_shot = first_shot.reset_index(name='percentage')
order_list = ['Han', 'Greedo']
q2chart1 = (alt.Chart(first_shot)
                .mark_bar()
                .encode(
                    x = alt.X('percentage', axis=alt.Axis(labels=False, format='%', title='')),
                    y = alt.Y('first_shot_character', axis=alt.Axis(title=''), sort=order_list), text='
                )

)
q2chart1_1 = ((alt.Chart(first_shot))
            .mark_text(align='left', baseline='middle', dy=-3)
            .encode(
                    x = alt.X('percentage', axis=alt.Axis(labels=False, format='%', title='')),
                    y = alt.Y('first_shot_character', axis=alt.Axis(title=''), sort=order_list), text='
                )
            )
q2chart1f = (q2chart1 + q2chart1_1).properties(
                    title={
                    "text": "Who Shot First?",
                    "subtitle": "According to 834 respondents",
                    }).configure_title(anchor='start').configure_axis(grid=False)
q2chart1f.save('question_2_3.png')
# %%
# Grand Question 2: chart for 'which star wars movies have you seen'
# drop people who have seen_any but have all NaNs for all movies
watched = df_rows.filter(['seen_I_the_phantom_menace',
       'seen_II_attack_of_the_clones', 'seen_III_revenge_of_the_sith',
       'seen_IV_a_new_hope', 'seen_V_the_empire_strikes_back',
       'seen_VI_return_of_the_jedi']).dropna(how='all')
# check if the respondent number matches up the article: 835
watched.shape
watched2 = watched.notnull().sum() / watched.shape[0] * 100
watched2 = watched2.round(0)
# rename the second columns to percents
order_list2 = ['The Phantom Menace','Attack of the Clones','Revenge of the Sith','A New Hope', 'The Emp
watched2 = (watched2.reset_index(name = 'percentage')
            # rename indexs to movie names
            .replace('seen_I_the_phantom_menace','The Phantom Menace')
            .replace('seen_II_attack_of_the_clones','Attack of the Clones')
            .replace('seen_III_revenge_of_the_sith','Revenge of the Sith')
            .replace('seen_IV_a_new_hope','A New Hope')
            .replace('seen_V_the_empire_strikes_back','The Empire Strikes Back')
            .replace('seen_VI_return_of_the_jedi','Return of The Jedi'))
q2chart2 = (alt.Chart(watched2)
        .mark_bar()
        .encode(
            x = alt.X('percentage', axis=alt.Axis(labels=False,title='')),
            y = alt.Y('index', axis=alt.Axis(title=''), sort=order_list2), text='percentage'
        ))
q2chart2_1 = (alt.Chart(watched2)
                .mark_text(align='left', baseline='middle', dy=-3)
                .encode(
                    x = alt.X('percentage', axis=alt.Axis(labels=False,title='')),
                    y = alt.Y('index', axis=alt.Axis(title=''), sort=order_list2), text='percentage'
                )

)
q2chart2f = (q2chart2 + q2chart2_1).properties(
                    title={
                    "text": "Which 'Star Wars' Movies Have You Seen? ",
                    "subtitle": "of 835 respondents who have seen any film",
                    }).configure_title(anchor='start').configure_axis(grid=False)
q2chart2f.save('question_2_4.png')
'''
Another way to recreate chart 2
# move multiple columns and their values into one column
watched.melt().value.value_counts(sort=False)
'''


#%%
# Grand Question 3:
# a. Filter the dataset to respondents that have seen at least one film.
q3 = df_rows.query('seen_any == "Yes"')

# b. Create a new column that converts the age ranges to a single number.
# Drop the age range categorical column.
q3.age.str.split('-', expand=True).replace('> 60', '60').astype('float')
# (q3.age.str.replace first and str,split next will work too. .str doesn't like to work with more than
ml_age = (q3.age
    .str.split('-', expand=True)
    .replace('> 60', '60')
    .astype('float')
    # change the column name
    .rename(columns = {0:'age_min', 1:'age_max'})
    # drop age_max
    .age_min
)

# c. Create a new column that converts the school groupings to a single number.
# Drop the school categorical column.
ml_education = (q3.education
                    .replace('Less than high school degree', 10)
                    .replace('High school degree', 12)
                    .replace('Some college or Associate degree', 14)
                    .replace('Bachelor degree', 16)
                    .replace('Graduate degree', 20)
                )

# d. Create a new column that converts the income ranges to a single number.
# Drop the income range categorical column.
# same with age
ml_income = (q3.income
    .str.replace('\$|,|\+', '').str.split('-', expand=True)
    .astype('float')
    # change the column name
    .rename(columns = {0:'income_min', 1:'income_max'})
    # drop income_max
    .income_min
)

# e. One-hot encode (get_dummies) all remaining categorical columns. It can be done on single column an
onehot_ml = pd.get_dummies(q3.filter(['seen_any', 'is_fan', 'seen_I_the_phantom_menace',
       'seen_II_attack_of_the_clones', 'seen_III_revenge_of_the_sith',
       'seen_IV_a_new_hope', 'seen_V_the_empire_strikes_back',
       'seen_VI_return_of_the_jedi', 'rank_I_the_phantom_menace',
       'rank_II_attack_of_the_clones', 'rank_III_revenge_of_the_sith',
       'rank_IV_a_new_hope', 'rank_V_the_empire_strikes_back',
       'rank_VI_return_of_the_jedi', 'rate_han', 'rate_luke', 'rate_leia',
       'rate_skywalker', 'rate_obi', 'rate_palpatine', 'rate_vader',
       'rate_lando', 'rate_fett', 'rate_c3po', 'rate_r2d2', 'rate_jar',
       'rate_amidala', 'rate_yoda', 'first_shot_character', 'know_universe',
       'is_universe_fanŒ', 'is_franchise_fan', 'gender', 'region']))

# f. Create your target (also known as "y" or "label") column based on the new income range column.
y = ml_income > 50000

# put everything together
# use concat: axis=1 means to keep them side by side rather than by default axis=0 to stack rows all to
# concat functions only works for the same number of rows
# remember to drop income column
starwars_ml = (pd.concat([onehot_ml,
                        ml_age, ml_education, ml_income],
                        axis=1)).dropna()
# %%
# Grand Question 4:
# Split data
# x: features, y: target
x = starwars_ml.drop(columns='income_min')
y = starwars_ml.income_min > 50000
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=.3, random_state=76)

# Create model
classifier = RandomForestClassifier()

# Train model
classifier.fit(x_train, y_train)

# Make predictions
y_predictions = classifier.predict(x_test)

# assess classifier performance: accuracy
accuracy = metrics.accuracy_score(y_test, y_predictions)
accuracy = accuracy.round(2)
```