Client Report - The war with Star Wars

Course DS 250
Conner Crook

Elevator pitch

Data can be very hard to interpret and read when it is untidy. When you tidy your data and make it more readable, it is easier to predict outcomes from the data.

GRAND QUESTION 1

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

In the below code, I changed all the long names of the code to shorter names that we could more easily use with panda. I displayed the new 'seen_any' column to show that this was done.

TECHNICAL DETAILS

```
column_replace = {
    'Which of the following Star Wars films have you seen\\? Please select all that apply\\.':'s
    'Please rank the Star Wars films in order of preference with 1 being your favorite film in t
    'Please state whether you view the following characters favorably, unfavorably, or are unfam
    'Do you consider yourself to be a fan of the Star Trek franchise\\?':'star_trek_fan',
    'Do you consider yourself to be a fan of the Expanded Universe\\?\x8cæ':'expanded_fan',
    'Are you familiar with the Expanded Universe\\?':'know_expanded',
    'Have you seen any of the 6 films in the Star Wars franchise\\?':'seen_any',
    'Do you consider yourself to be a fan of the Star Wars film franchise\\?':'star_wars_fan',
    'Which character shot first\\?':'shot_first',
    'Unnamed: \d{1,2}':np.nan,
    ' ':'_',
}

response_replace = {
    'Response':'',
    'Star Wars: Episode ':'',
    ' ':'_'
}

#%%
# Replace the columns with the new names.
new_columns = (SW_cols
    .assign(
        value_replace = lambda x:  x.value.str.strip().replace(response_replace, regex=True),
        variable_replace = lambda x: x.variable.str.strip().replace(column_replace, regex=True)
    )
    .fillna(method = 'ffill')
    .fillna(value = "")
    .assign(column_names = lambda x: x.variable_replace.str.cat(x.value_replace, sep = "__").str
    )
#%%
# Merge new columns into the Star Wars dataset
SW_data.columns = new_columns.column_names.to_list()


print(SW_data
    .head(5)
    .filter(['seen_any'])
    .to_markdown(index=False))
```

seen_any
Yes
No
Yes

seen_any
Yes
Yes

GRAND QUESTION 2

Filter the dataset to those that have seen at least one film.

This one, I simply took the 'seen_any' column and filtered it to only show those who answered that they had seen at least one Star Wars film.

TECHNICAL DETAILS

```
SW_fil_data = (SW_data
    .query('seen_any == "Yes"'))


print(SW_fil_data
    .head(20)
    .filter(['respondentid', 'seen_any'])
    .to_markdown(index = False))
```

respondentid	seen_any
3292879998	Yes
3292765271	Yes
3292763116	Yes
3292731220	Yes
3292719380	Yes
3292684787	Yes
3292663732	Yes
3292654043	Yes
3292640424	Yes
3292637870	Yes

respondentid	seen_any
3292609214	Yes
3292596911	Yes
3292587240	Yes
3292583038	Yes
3292580516	Yes
3292572872	Yes
3292565282	Yes
3292562297	Yes
3292522349	Yes
3292521066	Yes

GRAND QUESTION 3

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.

I was unable to recreate a chart but the code below shows that the article is slightly different than the results that we have. I took the filtered data and found that there was a total of 936 people who answered that they had seen at least one film. The article states that there were 835 people who had seen a movie. The percentages are also slightly lower.

TECHNICAL DETAILS

```
chart_data = SW_fil_data.filter(['seen__i__the_phantom_menace', 'seen__ii__attack_of_the_clones'

chart_data = pd.get_dummies(chart_data, drop_first=False)
#%%
chart_data2 = chart_data.melt()
#%%
seen_movies = (chart_data2.groupby(['variable']).sum())

seen_movies = (seen_movies
    .assign(
        percentage = lambda x: (x.value / 936) * 100
    ))


print(seen_movies
    .to_markdown(index=True))
```

variable	value	percentage
seen__i__the_phantom_menace_Star Wars: Episode I The Phantom Menace	673	0.719017
seen__ii__attack_of_the_clones_Star Wars: Episode II Attack of the Clones	571	0.610043
seen__iii__revenge_of_the_sith_Star Wars: Episode III Revenge of the Sith	550	0.587607
seen__iv__a_new_hope_Star Wars: Episode IV A New Hope	607	0.648504
seen__v_the_empire_strikes_back_Star Wars: Episode V The Empire Strikes Back	758	0.809829
seen__vi_return_of_the_jedi_Star Wars: Episode VI Return of the Jedi	738	0.788462

GRAND QUESTION 4

Clean and format the data so that it can be used in a machine learning model. Please achieve the following requests and provide examples of the table with a short description the changes made in your report.

I changed all the required columns to floats. The rest of the categorical columns I One-hot encoded to change to boolean to use for machine learning. The below code and chart prove that this is done.

TECHNICAL DETAILS

```
age_num = (SW_fil_data.age.
        str.split("-", expand = True).
        rename(columns = {0: 'age_min', 1: 'age_max'}).
        apply(lambda x: x.str.replace("\>|,|\+", "")).
        astype('float'))

SW_fil_data = pd.concat([
    SW_fil_data,
    age_num.age_min
], axis = 1)

SW_fil_data.pop('age')
#%%
"""
Create an additional column that converts the school groupings to a
number and drop the school categorical column.
"""
education_level = (SW_fil_data.education
        .str.replace('Less than high school degree', '9')
        .str.replace('High school degree', '12')
        .str.replace('Some college or Associate degree', '14')
        .str.replace('Bachelor degree', '16')
        .str.replace('Graduate degree', '20')
        .astype('float'))

SW_fil_data.pop('education')

SW_fil_data = pd.concat([
    SW_fil_data,
    education_level
], axis = 1)

#%%
"""
Create an additional column that converts the income
ranges to a number and drop the income range categorical column.
"""

income_num = (SW_fil_data.household_income.
        str.split("-", expand = True).
        rename(columns = {0: 'income_min', 1: 'income_max'}).
        apply(lambda x: x.str.replace("\$|,|\+", "")).
        astype('float'))

SW_fil_data = pd.concat([
    SW_fil_data,
    income_num.income_min
], axis = 1)

SW_fil_data.pop('household_income')
```

```
#%%
"""
Create your target (also known as label) column based on the new income range column.
"""
target = SW_fil_data.filter(regex = 'income_min').fillna(0)
#%%
"""
One-hot encode all remaining categorical columns.
"""

final_sw = pd.get_dummies(SW_fil_data, drop_first=True)
```

	education	age_min	income_min
0	12	18	0
2	12	18	0
3	14	18	100000
4	14	18	100000
5	16	18	25000
6	12	18	0
7	12	18	0
8	14	18	0
9	14	18	25000
10	0	0	0
12	16	18	25000
13	20	30	50000
14	12	18	0
15	14	18	0
16	16	18	50000
17	14	18	0
18	12	18	0
19	14	18	50000

	education	age_min	income_min
20	14	18	0
21	16	18	0

GRAND QUESTION 5

Build a machine learning model that predicts whether a person makes more than $50k.

I used the RandomForestClassifier to create the model to predict the income of the respondent. I removed respondentid from the dataset while creating the model as I did not think it relevant in the prediction. The test is not very accurate.

TECHNICAL DETAILS

```
features = final_sw.drop(final_sw.filter(regex = 'income_min|respondentid').columns, axis = 1).f
feature_train, feature_test, target_train, target_test = train_test_split(features, target, test

#%%
# Use the RandomForestclassifier
classifier = RandomForestClassifier()

classifier.fit(feature_train, target_train)
#%%
# Predict using classifier
targets_predicted = classifier.predict(feature_test)
#%%
```

APPENDIX A (PYTHON CODE)

```
#%%
import pandas as pd
import altair as alt
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn import tree
from sklearn import metrics
#from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

#%%
url = 'https://github.com/fivethirtyeight/data/raw/master/star-wars-survey/StarWars.csv'

#%%
#Pull in Star Wars data and make it longer showing variable and values. Turned Columns to Rows
SW_cols = pd.read_csv(url, encoding = "ISO-8859-1", engine='python', nrows = 1).melt()

#%%
#Shows all the data pulled in from URL in its wide version
SW_data = pd.read_csv(url, encoding = "ISO-8859-1" ,skiprows =2, header = None )
#%%
chart_data_import = pd.read_csv(url, encoding = "ISO-8859-1" , header = 1 )
#%%
## GRAND QUESTION 1
"""
Shorten the column names and clean them up for easier use with pandas.
"""
# Set new names for Columns and the options.
column_replace = {
    'Which of the following Star Wars films have you seen\\? Please select all that apply\\.':'s
    'Please rank the Star Wars films in order of preference with 1 being your favorite film in t
    'Please state whether you view the following characters favorably, unfavorably, or are unfam
    'Do you consider yourself to be a fan of the Star Trek franchise\\?':'star_trek_fan',
    'Do you consider yourself to be a fan of the Expanded Universe\\?\x8cæ':'expanded_fan',
    'Are you familiar with the Expanded Universe\\?':'know_expanded',
    'Have you seen any of the 6 films in the Star Wars franchise\\?':'seen_any',
    'Do you consider yourself to be a fan of the Star Wars film franchise\\?':'star_wars_fan',
    'Which character shot first\\?':'shot_first',
    'Unnamed: \d{1,2}':np.nan,
    ' ':'_',
}

response_replace = {
    'Response':'',
    'Star Wars: Episode ':'',
    ' ':'_'
}

#%%
```

```
# Replace the columns with the new names.
new_columns = (SW_cols
    .assign(
        value_replace = lambda x:  x.value.str.strip().replace(response_replace, regex=True),
        variable_replace = lambda x: x.variable.str.strip().replace(column_replace, regex=True)
    )
    .fillna(method = 'ffill')
    .fillna(value = "")
    .assign(column_names = lambda x: x.variable_replace.str.cat(x.value_replace, sep = "__").str
    )
#%%
# Merge new columns into the Star Wars dataset
SW_data.columns = new_columns.column_names.to_list()
#%%
print(SW_data
    .head(5)
    .filter(['seen_any'])
    .to_markdown(index=False))
#%%
## GRAND QUESTION 2

"""
Filter the dataset to those that have seen at least one film.
"""
SW_fil_data = (SW_data
    .query('seen_any == "Yes"'))

#%%
print(SW_fil_data
    .head(20)
    .filter(['respondentid', 'seen_any'])
    .to_markdown(index = False))
#%%
## GRAND QUESTION 3

"""
Please validate that the data provided on GitHub lines up with the article by recreating 2 of th
and calculating 2 summaries that they report in the article.
"""
chart_data = SW_fil_data.filter(['seen__i__the_phantom_menace', 'seen__ii__attack_of_the_clones'

chart_data = pd.get_dummies(chart_data, drop_first=False)
#%%
chart_data2 = chart_data.melt()
#%%
seen_movies = (chart_data2.groupby(['variable']).sum())

seen_movies = (seen_movies
    .assign(
        percentage = lambda x: (x.value / 936) * 100
    ))
```

```
#%%
print(seen_movies
    .to_markdown(index=True))
#%%
## GRAND QUESTION 4
"""
Clean and format the data so that it can be used in a machine learning model. Please achieve the
requests and provide examples of the table with a short description the changes made in your rep

Create an additional column that converts the age ranges to a number and drop the age range cate
"""
age_num = (SW_fil_data.age.
        str.split("-", expand = True).
        rename(columns = {0: 'age_min', 1: 'age_max'}).
        apply(lambda x: x.str.replace("\>|,|\+", "")).
        astype('float'))

SW_fil_data = pd.concat([
    SW_fil_data,
    age_num.age_min
], axis = 1)

SW_fil_data.pop('age')
#%%
"""
Create an additional column that converts the school groupings to a
number and drop the school categorical column.
"""
education_level = (SW_fil_data.education
        .str.replace('Less than high school degree', '9')
        .str.replace('High school degree', '12')
        .str.replace('Some college or Associate degree', '14')
        .str.replace('Bachelor degree', '16')
        .str.replace('Graduate degree', '20')
        .astype('float'))

SW_fil_data.pop('education')

SW_fil_data = pd.concat([
    SW_fil_data,
    education_level
], axis = 1)

#%%
"""
Create an additional column that converts the income
ranges to a number and drop the income range categorical column.
"""

income_num = (SW_fil_data.household_income.
        str.split("-", expand = True).
```

```
        rename(columns = {0: 'income_min', 1: 'income_max'}).
        apply(lambda x: x.str.replace("\$|,|\+", "")).
        astype('float'))

SW_fil_data = pd.concat([
    SW_fil_data,
    income_num.income_min
], axis = 1)

SW_fil_data.pop('household_income')
#%%
"""
Create your target (also known as label) column based on the new income range column.
"""
target = SW_fil_data.filter(regex = 'income_min').fillna(0)
#%%
"""
One-hot encode all remaining categorical columns.
"""

final_sw = pd.get_dummies(SW_fil_data, drop_first=True)

#%%
print(SW_fil_data
    .head(20)
    .filter(['education', 'age_min', 'income_min'])
    .fillna(0)
    .to_markdown())
#%%
## GRAND QUESTION 5
"""
Build a machine learning model that predicts whether a person makes more than $50k.
"""

features = final_sw.drop(final_sw.filter(regex = 'income_min|respondentid').columns, axis = 1).f
feature_train, feature_test, target_train, target_test = train_test_split(features, target, test

#%%
# Use the RandomForestclassifier
classifier = RandomForestClassifier()

classifier.fit(feature_train, target_train)
#%%
# Predict using classifier
targets_predicted = classifier.predict(feature_test)
#%%
```