Client Report - Project 5

Course DS 250 Marcel Pratikto

Elevator pitch

This project will analyze the relationship between Star Wars and its viewers.

GRAND QUESTION 1

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

I manually renamed all the columns and inserted it back to the dataframe.

TECHNICAL DETAILS

```
sw_data = pd.read_csv(url, encoding="ISO-8859-1", header=None, skiprows=2)
sw_data.columns = sw_column_names
print(sw_data.columns)
```

Index(['RespondentID', 'seen_SW', 'is_fan', 'seen_I', 'seen_II', 'seen_III', 'seen_IV', 'seen_V', 'seen_VI', 'rank_I', 'rank_II', 'rank_III', 'rank_IV', 'rank_V', 'rank_VI', 'Han Solo', 'Luke Skywalker', 'Princess Leia', 'Anakin', 'Obi Wan', 'Palpatine', 'Darth Vader', 'Lando', 'Boba Fett', 'C3P0', 'R2D2', 'Jar Jar', 'Padme', 'Yoda', 'shot_first', 'familiar_expanded_universe', 'fan_expanded_universe', 'fan_star_trek', 'Gender', 'Age', 'Household Income', 'Education', 'Location'], dtype='object')

GRAND QUESTION 2

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

Since the only way a person would have seen at least one star wars movie is if they answered yes to seen_SW, I decided to use that as the filter.

TECHNICAL DETAILS

```
sw_data = sw_data[sw_data["seen_SW"] == "Yes"]
```

+----------------+-----------+----------+ | RespondentID | seen_SW | is_fan | +================+===========+==========+ | 3292879998 | Yes | Yes | +----------------+-----------+----------+ | 3292765271 | Yes | No | +----------------+-----------+----------+

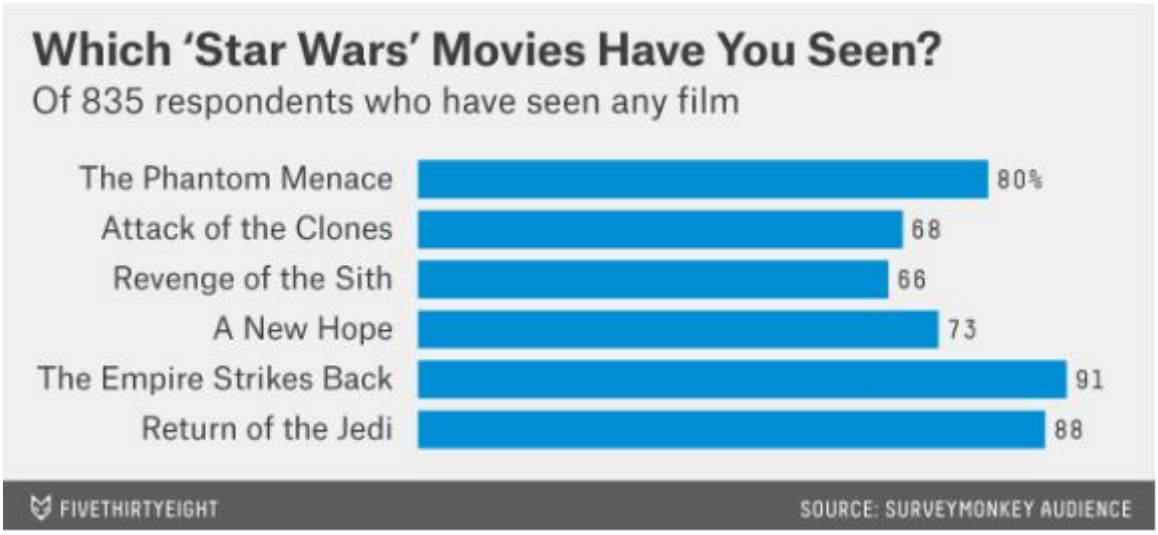
GRAND QUESTION 3

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.

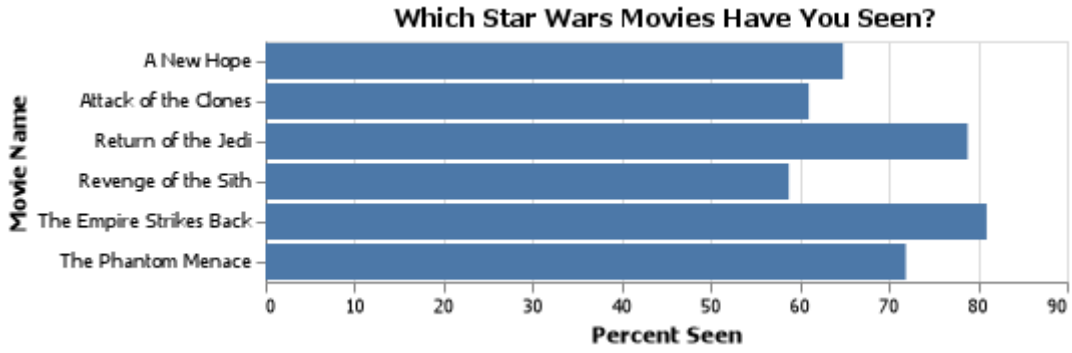
The first visual compares percentages of the 835 respondents who has seen at least 1 Star Wars movie to each of the Star Wars movies. The second visual shows the percentages of which movie is their favorite. This one is limited to only 471 respondents who has seen all six Star Wars movies. To recreate the original visual, I had to do some data cleaning: movies seen, movie rankings.

TECHNICAL DETAILS

Visual 1 Original



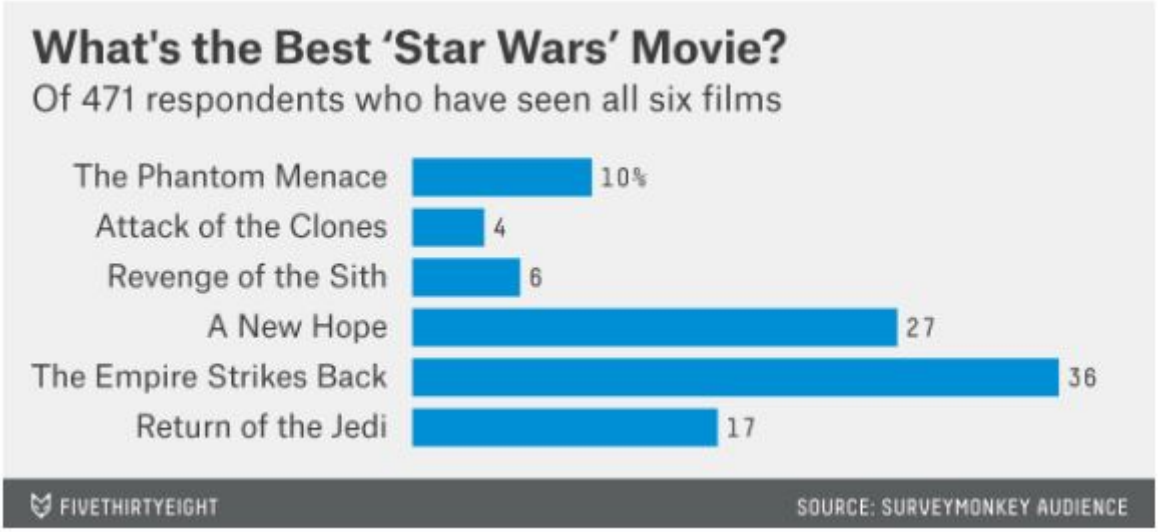
Visual 1



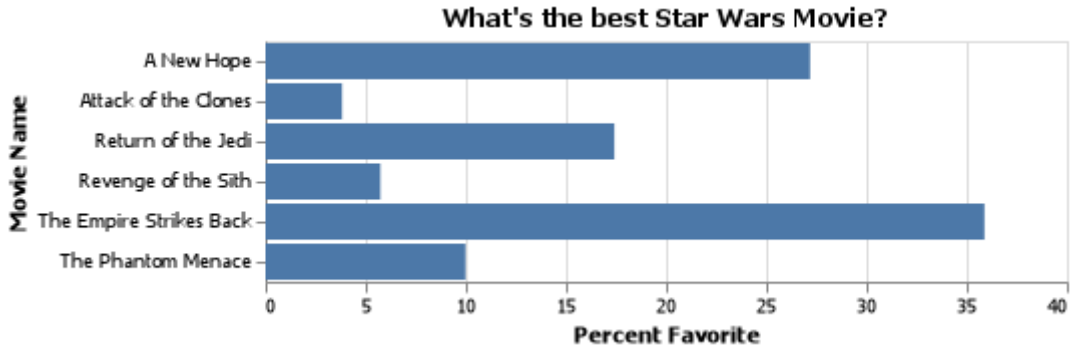
Recreation

Visual 2

Original



Visual



2 Recreation

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 of the changes made in your report.

Create an additional column that converts the age ranges to a number and drop the age range categorical column.

I converted the age ranges to numbers. 18-29 becomes 1, 30-44 becomes 2, 45-60 becomes 3, >60 becomes 4, and anything else is 0.

```
# Create an additional column that converts the age ranges to a number and drop
the age range categorical column.
new_age = []
for age in sw_data["Age"]:
    if age == "18-29":
        age = 1
    elif age == "30-44":
        age = 2
    elif age == "45-60":
        age = 3
    elif age == "> 60":
        age = 4
    else:
        age = 0
```

```
    new_age.append(age)
sw_data["Age"] = new_age
sw_data["Age"]
```

index Age 1180 3 1181 1 1182 2 1184 3 1185 4

Create an additional column that converts the school groupings to a number and drop the school categorical column.

I converted the school groupings into numbers. High school is 1, Some college/Associate is 2, Bachelor is 3, Graduate is 4, and anything else is 0.

```
# Create an additional column that converts the school groupings to a number and
drop the school categorical column.
new_education = []
for degree in sw_data["Education"]:
    if degree == "High school degree":
        degree = 1
    elif degree == "Some college or Associate degree":
        degree = 2
    elif degree == "Bachelor degree":
        degree = 3
    elif degree == "Graduate degree":
        degree = 4
    else:
        degree = 0
    new_education.append(degree)
sw_data["Education"] = new_education
sw_data["Education"]
```

index Education 1180 2 1181 2 1182 3 1184 2 1185 4

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

I converted the income ranges into numbers. 0-24,999 is 1, 25,000-49,999 is 2, 50,000-99,999 is 3, 100,000-149,999 is 4, 150,000+ is 5, and anything else is 0.

```
# Create an additional column that converts the income ranges to a number and drop
the income range categorical column.
new_income = []
for income in sw_data["Household Income"]:
    if income == "$0 - $24,999":
        income = 1
    elif income == "$25,000 - $49,999":
        income = 2
    elif income == "$50,000 - $99,999":
        income = 3
```

```
    elif income == "$100,000 - $149,999":
        income = 4
    elif income == "$150,000+":
        income = 5
    else:
        income = 0
    new_income.append(income)
sw_data["Household Income"] = new_income
sw_data["Household Income"]
```

index Household Income 1180 1 1181 1 1182 3 1184 4 1185 3

Create your target (also known as label) column based on the new income range column.

The target is based on the income. It displays a true if the person makes $50,000 or more.

```
# Create your target (also known as label) column based on the new income range
column.
over_50k = []
for income in sw_data["Household Income"]:
    if income >= 3:
        over_50k.append(True)
    else:
        over_50k.append(False)
sw_data["over_50k"] = over_50k
sw_data["over_50k"]
```

index over_50k 1180 False 1181 False 1182 True 1184 True 1185 True

One-hot encode all remaining categorical columns.

I applied the same logic to the remaining categorical columns. See appendix for the full code.

GRAND QUESTION 5

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

I was able to build a machine learning model that predicts whether a person makes more than $50k with ~62% accuracy. I used "is_fan", "fan_star_trek", "Gender", "Age", and "Education" columns as the data input and the column "over_50k" as the target.

TECHNICAL DETAILS

```
sw_data.columns
x = sw_data.filter(["is_fan", "fan_star_trek", "Gender", "Age", "Education"])
y = sw_data["over_50k"]
print(sw_data.shape)
print(x.shape)
```

```
print(y.shape)

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.081,
random_state = 864)
#create the model
classifier = RandomForestClassifier()
#train the model
classifier.fit(x_train, y_train)
#make predictions
y_predictions = classifier.predict(x_test)
#test how accurate predictions are
metrics.accuracy_score(y_test, y_predictions)
```

APPENDIX A (PYTHON CODE)

```
import pandas as pd
import altair as alt
import numpy as np

url = "StarWars.csv"

""" GRAND QUESTION 1
Shorten the column names and clean them up for easier use with pandas.
"""
#sw_columns = pd.read_csv(url, encoding="ISO-8859-1", header=None, nrows=2)
sw_column_names = [
    "RespondentID",
    "seen_SW",
    "is_fan",
    "seen_I",
    "seen_II",
    "seen_III",
    "seen_IV",
    "seen_V",
    "seen_VI",
    "rank_I",
    "rank_II",
    "rank_III",
    "rank_IV",
    "rank_V",
    "rank_VI",
    "Han Solo",
    "Luke Skywalker",
    "Princess Leia",
    "Anakin",
    "Obi Wan",
    "Palpatine",
    "Darth Vader",
    "Lando",
    "Boba Fett",
    "C3P0",
```

```
    "R2D2",
    "Jar Jar",
    "Padme",
    "Yoda",
    "shot_first",
    "familiar_expanded_universe",
    "fan_expanded_universe",
    "fan_star_trek",
    "Gender",
    "Age",
    "Household Income",
    "Education",
    "Location"
]
sw_data = pd.read_csv(url, encoding="ISO-8859-1", header=None, skiprows=2)
sw_data.columns = sw_column_names
#sw_data.head()
#print(sw_data.head(1).to_markdown(index=False, tablefmt="grid"))
print(sw_data.columns)

""" GRAND QUESTION 2
Filter the dataset to those that have seen at least one film.
"""
sw_data = sw_data[sw_data["seen_SW"] == "Yes"]
sw_data.head()
print(sw_data.head().to_markdown(index=False, tablefmt="grid"))

""" 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.
"""
# clean up data for visual 1
sw_data["seen_I"] = (sw_data["seen_I"]
    .replace("Star Wars: Episode I  The Phantom Menace", 1)
    .replace(np.nan, 0)
)
sw_data["seen_II"] = (sw_data["seen_II"]
    .replace("Star Wars: Episode II  Attack of the Clones", 1)
    .replace(np.nan, 0)
)
sw_data["seen_III"] = (sw_data["seen_III"]
    .replace("Star Wars: Episode III  Revenge of the Sith", 1)
    .replace(np.nan, 0)
)
sw_data["seen_IV"] = (sw_data["seen_IV"]
    .replace("Star Wars: Episode IV  A New Hope", 1)
    .replace(np.nan, 0)
)
sw_data["seen_V"] = (sw_data["seen_V"]
    .replace("Star Wars: Episode V The Empire Strikes Back", 1)
    .replace(np.nan, 0)
)
sw_data["seen_VI"] = (sw_data["seen_VI"]
```

```
    .replace("Star Wars: Episode VI Return of the Jedi", 1)
    .replace(np.nan, 0)
)
sw_data.head()
# total number of respondents who has seen at least 1 movie
respondents_seen = sw_data["seen_SW"].count()
respondents_seen
seen_d = {
    "name": ["The Phantom Menace", "Attack of the Clones", "Revenge of the Sith",
"A New Hope", "The Empire Strikes Back", "Return of the Jedi"],
    "percent_seen": []
}
seen_d["percent_seen"].append(sw_data["seen_I"].sum()/respondents_seen*100)
seen_d["percent_seen"].append(sw_data["seen_II"].sum()/respondents_seen*100)
seen_d["percent_seen"].append(sw_data["seen_III"].sum()/respondents_seen*100)
seen_d["percent_seen"].append(sw_data["seen_IV"].sum()/respondents_seen*100)
seen_d["percent_seen"].append(sw_data["seen_V"].sum()/respondents_seen*100)
seen_d["percent_seen"].append(sw_data["seen_VI"].sum()/respondents_seen*100)
seen_df = pd.DataFrame(data=seen_d)
seen_df
visual_1 = (alt.Chart(seen_df)
    .mark_bar()
    .encode(
        x = alt.X("percent_seen", axis=alt.Axis(title="Percent Seen")),
        y = alt.Y("name", axis=alt.Axis(title="Movie Name"))
    )
    .properties(title="Which Star Wars Movies Have You Seen?")
)
visual_1
visual_1.save("visual_1.png")

# Visual 2
favorite_d = {
    "name": ["The Phantom Menace", "Attack of the Clones", "Revenge of the Sith",
"A New Hope", "The Empire Strikes Back", "Return of the Jedi"],
    "percent_fav": [0,0,0,0,0,0],
    "count_fav": [0,0,0,0,0,0]
}

count = 0
for i in range(len(sw_data["seen_SW"])):
    data = sw_data.iloc[i]
    if data["seen_I"] + data["seen_II"] + data["seen_III"] + data["seen_IV"] +
data["seen_V"] + data["seen_VI"] == 6.0:
        count += 1
        if data["rank_I"] == 1.0:
            favorite_d["count_fav"][0] += 1
        elif data["rank_II"] == 1.0:
            favorite_d["count_fav"][1] += 1
        elif data["rank_III"] == 1.0:
            favorite_d["count_fav"][2] += 1
        elif data["rank_IV"] == 1.0:
            favorite_d["count_fav"][3] += 1
        elif data["rank_V"] == 1.0:
```

```
            favorite_d["count_fav"][4] += 1
        elif data["rank_VI"] == 1.0:
            favorite_d["count_fav"][5] += 1

for i in range(6):
    favorite_d["percent_fav"][i] = favorite_d["count_fav"][i]/count*100

favorite_df = pd.DataFrame(data=favorite_d)
favorite_df

visual_2 = (alt.Chart(favorite_df)
    .mark_bar()
    .encode(
        x = alt.X("percent_fav", axis=alt.Axis(title="Percent Favorite")),
        y = alt.Y("name", axis=alt.Axis(title="Movie Name"))
    )
    .properties(title="What's the best Star Wars Movie?")
)
visual_2
visual_2.save("visual_2.png")

""" 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.

Create an additional column that converts the age ranges to a number and drop the
age range categorical column.
Create an additional column that converts the school groupings to a number and
drop the school categorical column.
Create an additional column that converts the income ranges to a number and drop
the income range categorical column.
Create your target (also known as label) column based on the new income range
column.
One-hot encode all remaining categorical columns.
"""
# age category
ages = []
for age in sw_data["Age"]:
    if age not in ages:
        ages.append(age)
ages
# Create an additional column that converts the age ranges to a number and drop
the age range categorical column.
new_age = []
for age in sw_data["Age"]:
    if age == "18-29":
        age = 1
    elif age == "30-44":
        age = 2
    elif age == "45-60":
        age = 3
    elif age == "> 60":
        age = 4
```

```
    else:
        age = 0
    new_age.append(age)
sw_data["Age"] = new_age
sw_data["Age"]

# school category
degrees = []
for degree in sw_data["Education"]:
    if degree not in degrees:
        degrees.append(degree)
degrees
# Create an additional column that converts the school groupings to a number and
drop the school categorical column.
new_education = []
for degree in sw_data["Education"]:
    if degree == "High school degree":
        degree = 1
    elif degree == "Some college or Associate degree":
        degree = 2
    elif degree == "Bachelor degree":
        degree = 3
    elif degree == "Graduate degree":
        degree = 4
    else:
        degree = 0
    new_education.append(degree)
sw_data["Education"] = new_education
sw_data["Education"]

# income category
incomes = []
for income in sw_data["Household Income"]:
    if income not in incomes:
        incomes.append(income)
incomes
# Create an additional column that converts the income ranges to a number and drop
the income range categorical column.
new_income = []
for income in sw_data["Household Income"]:
    if income == "$0 - $24,999":
        income = 1
    elif income == "$25,000 - $49,999":
        income = 2
    elif income == "$50,000 - $99,999":
        income = 3
    elif income == "$100,000 - $149,999":
        income = 4
    elif income == "$150,000+":
        income = 5
    else:
        income = 0
    new_income.append(income)
sw_data["Household Income"] = new_income
```

```
sw_data["Household Income"]

# Create your target (also known as label) column based on the new income range
column.
over_50k = []
for income in sw_data["Household Income"]:
    if income >= 3:
        over_50k.append(True)
    else:
        over_50k.append(False)
sw_data["over_50k"] = over_50k
sw_data["over_50k"]

# One-hot encode all remaining categorical columns.
sw_data.columns
sw_data.head()

new_seen_SW = []
for seen in sw_data["seen_SW"]:
    if seen == "Yes":
        new_seen_SW.append(True)
    else:
        new_seen_SW.append(False)
sw_data["seen_SW"] = new_seen_SW
sw_data["seen_SW"]

new_fan = []
for fan in sw_data["is_fan"]:
    if fan == "Yes":
        new_fan.append(True)
    else:
        new_fan.append(False)
sw_data["is_fan"] = new_fan
sw_data["is_fan"]

shots = []
for shot in sw_data["shot_first"]:
    if shot not in shots:
        shots.append(shot)
shots
new_shot_first = []
for shot in sw_data["shot_first"]:
    if shot == "Greedo":
        shot = 1
    elif shot == "Han":
        shot = 2
    else:
        shot = 0
    new_shot_first.append(shot)
sw_data["shot_first"] = new_shot_first
sw_data["shot_first"]

new_familiar_expanded_universe = []
for familiar in sw_data["familiar_expanded_universe"]:
```

```
    if familiar == "Yes":
        new_familiar_expanded_universe.append(True)
    else:
        new_familiar_expanded_universe.append(False)
sw_data["familiar_expanded_universe"] = new_familiar_expanded_universe
sw_data["familiar_expanded_universe"]

new_fan_expanded_universe = []
for fan in sw_data["fan_expanded_universe"]:
    if fan == "Yes":
        new_fan_expanded_universe.append(True)
    else:
        new_fan_expanded_universe.append(False)
sw_data["fan_expanded_universe"] = new_fan_expanded_universe
sw_data["fan_expanded_universe"]

new_fan_star_trek = []
for fan in sw_data["fan_star_trek"]:
    if fan == "Yes":
        new_fan_star_trek.append(True)
    else:
        new_fan_star_trek.append(False)
sw_data["fan_star_trek"] = new_fan_star_trek
sw_data["fan_star_trek"]

genders = []
for gender in sw_data["Gender"]:
    if gender not in genders:
        genders.append(gender)
genders
new_gender = []
for gender in sw_data["Gender"]:
    if gender == "Male":
        gender = 1
    elif gender == "Female":
        gender = 2
    else:
        gender = 0
    new_gender.append(gender)
sw_data["Gender"] = new_gender
sw_data["Gender"]

""" GRAND QUESTION 5
Build a machine learning model that predicts whether a person makes more than
$50k.
"""
# imports
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
```

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import plot_confusion_matrix

sw_data.columns
x = sw_data.filter(["is_fan", "fan_star_trek", "Gender", "Age", "Education"])
y = sw_data["over_50k"]
print(sw_data.shape)
print(x.shape)
print(y.shape)

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.081,
random_state = 864)
#create the model
classifier = RandomForestClassifier()
#train the model
classifier.fit(x_train, y_train)
#make predictions
y_predictions = classifier.predict(x_test)
#test how accurate predictions are
metrics.accuracy_score(y_test, y_predictions)
```