Winning Jeopardy - apply Chi-square test

by Susan Fisher

If you're going to be a contestant on the Jeopardy game show, you might wonder whether it is beneficial to study past questions or to not study them at all. This project quantitatively shows that performance can be improved by studying past questions.

A focus on studying questions that are high money value versus low money value could be a way to make more money on the game show. The chi-square test statistic is used to determine whether there is a significant difference in word usage for high value questions versus low value ones.

The dataset contains 216,930 rows of questions, answers, and money value. For this project, the data will be reduced to 10,000 rows so the code will compile in a timely fashion.

The following website is where the data can be downloaded and additional information can be found: https://www.reddit.com/r/datasets/comments/1uyd0t/200000_jeopardy_questions_in_a_json_file/

In [1]:

```
# Read in the dataset and import numpy and pandas libraries

import numpy as np
import pandas as pd

raw_data = pd.read_csv('C:/Users/Name/Documents/Python Scripts/DataSets/JEOPARDY_CSV.csv')
```

In [2]:

```
# Reduce dataset from 219,930 by taking 5% to be used for this project.
# The full dataset will take more time to compile the code.

# Take a random sample of 5% of the original dataset
n_rows = round(len(raw_data) * 0.05)
jeopardy =  raw_data.sample(n=n_rows, random_state=1, axis=0).reset_index(drop=True)

jeopardy.shape
```

Out[2]:

```
(10846, 7)
```

DATA EXPLORATION

In [3]:

```
jeopardy.head(3)
```

Out[3]:

	Show Number	Air Date	Round	Category	Value	Question	Answer
0	5842	2010-01-26	Double Jeopardy!	OF DISCIPLINE	$2000	This type of yoga is Sanskrit for "discipline ...	hatha yoga
1	1322	1990-05-08	Jeopardy!	HISTORY	$300	4 treaties to mitigate the horrors of war were...	Geneva
2	4136	2002-09-02	Double Jeopardy!	EYE ON ASIA	$400	On Dec. 13, 1937 Japan took over the city of N...	China

In [4]:

```
jeopardy.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10846 entries, 0 to 10845
Data columns (total 7 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   Show Number  10846 non-null  int64 
 1    Air Date    10846 non-null  object
 2    Round       10846 non-null  object
 3    Category    10846 non-null  object
 4    Value       10846 non-null  object
 5    Question    10846 non-null  object
 6    Answer      10846 non-null  object
dtypes: int64(1), object(6)
memory usage: 593.3+ KB
```

In [5]:

```
jeopardy.shape
```

Out[5]:

```
(10846, 7)
```

DATA CLEANING

For data analysis, the following data cleaning is needed: 1) Column Names: remove leading space
2) 'Question' and 'Answer' columns: remove punctuation, and convert all words to lowercase
3) 'Value' column: remove '$', and convert to numeric type
4) 'Air Date' column: convert to datetime type

(1) Data Cleaning - Column Names

> **Remove leading space**

In [6]:

```
# Column names: before removing leading space
jeopardy.columns
```

Out[6]:

```
Index(['Show Number', ' Air Date', ' Round', ' Category', ' Value',
       ' Question', ' Answer'],
      dtype='object')
```

In [7]:

```
# Column names: remove leading space
jeopardy.columns = jeopardy.columns.str.replace('^\s', '')

jeopardy.columns
```

Out[7]:

```
Index(['Show Number', 'Air Date', 'Round', 'Category', 'Value', 'Question',
       'Answer'],
      dtype='object')
```

(2) Data Cleaning - 'Question' and 'Answer' columns

> **Remove punctuation, and convert all words to lowercase.**

In [8]:

```
# 'Question' and 'Answer' columns: before data cleaning
print(jeopardy['Question'].head())
print('\n')
print(jeopardy['Answer'].head())
```

```
0    This type of yoga is Sanskrit for "discipline ...
1    4 treaties to mitigate the horrors of war were...
2    On Dec. 13, 1937 Japan took over the city of N...
3    It's the island where Fay Wray first encounter...
4    The Metropolitan Museum of Art paid a record $...
Name: Question, dtype: object


0       hatha yoga
1           Geneva
2            China
3     Skull Island
4    Playing cards
Name: Answer, dtype: object
```

In [9]:

```
# 'Question' and 'Answer' columns:
     # convert all letters to lowercase
     # removes all punctuation, except white spaces

jeopardy['clean_question'] = jeopardy['Question'].str.replace('[^A-Za-z0-9\s]', '').s
tr.lower()
jeopardy['clean_answer'] = jeopardy['Answer'].str.replace('[^A-Za-z0-9\s]', '').str.l
ower()

print(jeopardy['clean_question'].head())
print('\n')
print(jeopardy['clean_answer'].head())
```

```
0    this type of yoga is sanskrit for discipline o...
1    4 treaties to mitigate the horrors of war were...
2    on dec 13 1937 japan took over the city of nan...
3    its the island where fay wray first encountere...
4    the metropolitan museum of art paid a record 1...
Name: clean_question, dtype: object


0       hatha yoga
1           geneva
2            china
3     skull island
4    playing cards
Name: clean_answer, dtype: object
```

(3) Data Cleaning - 'Value' column

> **Remove '$', and convert to numeric type**

In [10]:

```
# 'Value' column: before cleaning
jeopardy['Value'].value_counts().tail(10)
```

Out[10]:

```
$9,000    1
$1,111    1
$8,600    1
$8,000    1
$4,800    1
$3,800    1
$4,008    1
$5,600    1
$7,800    1
$2,700    1
Name: Value, dtype: int64
```

In [11]:

```
# 'Value' column: remove '$', and convert to numeric type

jeopardy['clean_value'] = jeopardy['Value'].str.replace('[^A-Za-z0-9\s]', '')

jeopardy['clean_value'].unique()
```

Out[11]:

```
array(['2000', '300', '400', '100', '800', '3000', '600', '500', '200',
       '1000', '1200', 'None', '1600', '2500', '1400', '4000', '5000',
       '1500', '2200', '2600', '4400', '700', '3200', '1800', '2700',
       '3400', '8000', '10000', '4008', '1900', '2100', '5600', '7800',
       '1700', '2800', '1300', '3800', '900', '6000', '3600', '4800',
       '1100', '2400', '1111', '5400', '9000', '8600', '3500', '4200'],
      dtype=object)
```

In [12]:

```
# 'Value' column:  replace 'None' with '0', then convert to numeric type

jeopardy['clean_value'] = jeopardy['clean_value'].str.replace('None', '0').astype(int
)

print(jeopardy['clean_value'].dtype)
jeopardy['clean_value'].unique()
```

```
int32
```

Out[12]:

```
array([ 2000,   300,   400,   100,   800,  3000,   600,   500,   200,
        1000,  1200,     0,  1600,  2500,  1400,  4000,  5000,  1500,
        2200,  2600,  4400,   700,  3200,  1800,  2700,  3400,  8000,
       10000,  4008,  1900,  2100,  5600,  7800,  1700,  2800,  1300,
        3800,   900,  6000,  3600,  4800,  1100,  2400,  1111,  5400,
        9000,  8600,  3500,  4200])
```

(4) Data Cleaning - 'Air' column

> **Convert to datetime type**

In [13]:

```
jeopardy['Air Date'] = pd.to_datetime(jeopardy['Air Date'])
jeopardy['Air Date'].dtype
```

Out[13]:

```
dtype('<M8[ns]')
```

DATA ANALYSIS

The first part of the analysis will look at whether studying past questions can improve a contestant's performance on the game show, Jeopary. The second part of the analysis will try to answer if more money can be made by focusing on high money value questions versus low money value ones.

To decide on studying past questions, it would be helpful to know the following:

1. **how often the answer is deducible from the question**
2. **how often new questions are repeats of older questions**

Data Analysis - Study Past Questions

> **How often the answer is deducible from the question: find the number of times words in the answer also occur in the question.**

In [14]:

```
# Study Past Questions: how often the answer is deducible from the question.

# Create function, count_matches, that for every word in the answer, counts the match
es in the question.
# The function returns the frequency of occurrence of each word match in both the ans
wer and question.

def count_matches(row):
    match_count = 0

    # convert string to a list of strings
    split_question = row['clean_question'].split()
    split_answer = row['clean_answer'].split()

    # removes article, "the," since it is not relevant
    if "the" in split_answer:
            split_answer.remove('the')
    if len(split_answer) == 0:
            return 0

    for item in split_answer:
        if item in split_question:
            match_count += 1
    return (match_count / len(split_answer))

jeopardy['answer_in_question'] = jeopardy.apply(count_matches, axis=1)

# Mean of 'answer_in_question' column
jeopardy['answer_in_question'].mean()
```

Out[14]:

```
0.05959285633128524
```

The mean is only 6%, so there were few word matches in both the 'question' and 'answer.' This indicates that the answer is rarely deducible from the question.

Data Analysis - Study Past Questions

> **How often new questions are repeats of older questions: find frequency of how often complex words (> 5 characters) reoccur.**

In [15]:

```
# Study Past Questions: how often new questions are repeats of older questions
```

```
#question_overlap is a list of unique words that
question_overlap = []
terms_used = set()       #a set only retains unique words

jeopardy.sort_values('Air Date', inplace=True)

for index, row in jeopardy.iterrows():
    split_question = row['clean_question'].split()

    #Only keep words that are > 5 characters long
    split_question = [word for word in split_question if len(word) > 5]

    match_count = 0

    for word in split_question:
        if word in terms_used:
            match_count += 1
    for word in split_question:
        terms_used.add(word)

    if len(split_question) > 0:
        match_count /= len(split_question)

    question_overlap.append(match_count)

jeopardy['question_overlap'] = question_overlap

jeopardy['question_overlap'].mean()
```

Out[15]:

```
0.6164496056048508
```

The mean is about 70%, or 70% of certain words are repeated in past questions. Based on this alone, it appears that review of past questions is a good idea and will improve performance on the game show.

However, the data is only about 10% of the full jeopardy question dataset. And this analysis was for words with characters > 5, and not for phrases.

Data Analysis - High value questions vs Low value questions

Focusing on studying questions that are high money value rather than low value ones will help a contestant earn more on Jeopardy. Chi-squared test will be used to determine whether there is a significant difference in word usage for high value and low value questions.

The question categories are:

> **Low value questions: where 'Value' < $800**
>
> **High value questions: where 'Value' > $800**

In [16]:

```
# Function, high_values, categorizes questions as high value or low value.
# High value questions are assigned a "1," and low value ones are assigned a "0."

def high_values(df):
    if df['clean_value'] > 800:
        value = 1
    else:
        value = 0
    return value

jeopardy['high_value'] = jeopardy.apply(high_values, axis=1)
```

In [17]:

```
#Function, low_high_counts, counts the total number of high value and low value quest
ions

def low_high_counts(word):
    low_count = 0
    high_count = 0
    for index, row in jeopardy.iterrows():
        if word in row['clean_question'].split(" "):
            if row['high_value'] == 1:
                high_count += 1
            else:
                low_count += 1
    return low_count, high_count
```

In [18]:

```
# Randomly select 10 elements of the set of unique words in the questions, 'terms_use
d'

import random
comparison_terms = random.sample(terms_used, 10)

comparison_terms[:5]
```

Out[18]:

```
['effective', 'peanuts', 'proper', 'appeal', 'expreacher']
```

In [19]:

```
# Apply function, low_high_counts, to comparison_terms.
# For a word in comparison_terms, low_high_counts looks for the word in a question,
# then counts as high value or low value question

low_high_sample = []

for term in comparison_terms:
    low_high_sample.append(low_high_counts(term))

low_high_sample[:5]
```

Out[19]:

```
[(2, 0), (4, 0), (3, 1), (0, 3), (0, 1)]
```

Chi-square statistic test can be performed on the sample low/high value counts and the dataset low/high value counts. The observed data is the low/high counts taken from the random sample of ten unique words in the questions. The expected data is the low/high counts of the dataset.

Chi-square test is given by:

$$\chi^2 = \sum \frac{(Observed - Expected)}{Expected}$$

In [20]:

```
# Chi-squared test

from scipy.stats import chisquare

# For expected data: number of rows for high value count & low value count
high_value_count = jeopardy[jeopardy['high_value'] == 1].shape[0]
low_value_count = jeopardy[jeopardy['high_value'] == 0].shape[0]
```

```
chi_squared = []

for low_high in low_high_sample:
    total = sum(low_high)
    total_prop = total / jeopardy.shape[0]
    high_value_exp = total_prop * high_value_count
    low_value_exp = total_prop * low_value_count

    observed = np.array([low_high[0], low_high[1]])
    expected = np.array([high_value_exp, low_value_exp])
    chi_squared.append(chisquare(observed, expected))

chi_squared
```

Out[20]:

```
[Power_divergenceResult(statistic=4.892913886240865, pvalue=0.026967135266137608),
 Power_divergenceResult(statistic=9.78582777248173, pvalue=0.001758620419774108),
 Power_divergenceResult(statistic=4.10671671794252, pvalue=0.04271320729328355),
 Power_divergenceResult(statistic=1.226263151058579, pvalue=0.268134998995775),
 Power_divergenceResult(statistic=0.408754383686193, pvalue=0.522602408963224),
 Power_divergenceResult(statistic=0.408754383686193, pvalue=0.522602408963224),
 Power_divergenceResult(statistic=2.4464569431204324, pvalue=0.11779047312453814),
 Power_divergenceResult(statistic=0.027158159288401405, pvalue=0.8691035073005604),
 Power_divergenceResult(statistic=0.408754383686193, pvalue=0.522602408963224),
 Power_divergenceResult(statistic=2.3306260045657328, pvalue=0.12685085501566856)]
```

Of the ten sample points, the Chi-square statistic ranges from 4.9 to 13.3. Since the Chi-square test statistics are not large, then this indicates that the sample words did not have a statistically significant difference in usage between high value and low value questions (a very large Chi-square value would mean that there is a difference in usage).

CONCLUSION

This project looked at how to improve a contestant's performance on the game show, Jeopardy. In order to decide to study past questions or not to, first, an analysis of how often the question is deducible from the answer. It was only about 6%, so it's unlikely to come up with the question from the answer.

Second, an analysis of how often new questions are repeats of older questions. The result was about 70%, so studying questions can improve one's performance on the game show.

Lastly, Chi-square test was applied in order to determine if there is a significant difference in word usage for high value and low value questions. Of the ten sample points, it was determined that there was not a significant difference.

For future iterations of this project, some possible next steps are:

- **Eliminate more non-informative words such as "than," "a." Perhaps remove words that occur in more than a certain percentage of questions.**
- **In looking at whether questions are repeated, use phrases instead of individual words. Single words do not provide as accurate of analysis.**
- **Use the entire dataset rather than just 5% of the data, which this project used.**
- **Apply the chi-square test on more terms to determine which terms have larger differences. Only select terms that have high frequencies.**
- **Look at the "Category" column to determine if some categories occur more often. Studying could focus more on those categories.**