As always, I had no idea what I was doing when I started, but I learned so much and acutally put something together! I am proud of myself! Even though AI was helpful for the middle part of the work, it also gave me really good explanations for the work which helped me understand what I was doing. There were many attempts to reach this cleaned up notebook, but I would have never been here without so many mistakes!

Loading Libraries and such needed for the module's work

```
import json
import pandas as pd
import os
import numpy as np
import nltk
import sklearn
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_repor
```

```
nltk.download('punkt')
nltk.download('punkt_tab')
nltk.download('stopwords')
nltk.download('wordnet')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
True
```

2. Looking at the data

```
with open('jeopardy.json') as f:
    data = json.load(f)
```

```
# Convert the data to a Pandas DataFrame
df = pd.DataFrame(data)
df
```

	category	air_date	question	value	answer	
0	HISTORY	2004-12-31	'For the last 8 years of his life, Galileo was...	$200	Copernicus	Je
1	ESPN's TOP 10 ALL-TIME ATHLETES	2004-12-31	'No. 2: 1912 Olympian; football star at Carlis...	$200	Jim Thorpe	Je
2	EVERYBODY TALKS ABOUT IT...	2004-12-31	'The city of Yuma in this state has a record a...	$200	Arizona	Je
3	THE COMPANY LINE	2004-12-31	'In 1963, live on "The Art Linkletter Show", t...	$200	McDonald\'s	Je
4	EPITAPHS & TRIBUTES	2004-12-31	'Signer of the Dec. of Indep., framer of the C...	$200	John Adams	Je
...	...	...	...	...	...	
216925	RIDDLE ME THIS	2006-05-11	'This Puccini opera turns on the solution to 3...	$2000	Turandot	Je

Sarah Jackson
11:22 AM Today
(edited 11:23 AM Today)
Having trouble with this doing this all at the same time, so AI recommended the lambda

3. Prepping the data

Sarah Jackson
11:23 AM Today
I used unknown here to avoid errors. Not sure if that is the best choice, but it seemed to work.

Drop Categories

```
columns_to_drop = ['category', 'air_date', 'answer', 'show_numbe
df = df.drop(columns_to_drop, axis=1)
```

convert to lowercase

```
columns_to_lower = ['question', 'value', 'round']
for col in columns_to_lower:
    df[col] = df[col].apply(lambda x: x.lower() if isinstance(x, str) else x)
```

clean the missing values

```
df = df.fillna('Unknown')
```

Tokenize the words

```
    def tokenize_text(text):
        return nltk.word_tokenize(text)

    df['question_tokens'] = df['question'].apply(tokenize_text)
    df['value_tokens'] = df['value'].apply(tokenize_text)
    df['round_tokens'] = df['round'].apply(tokenize_text)
```

Lemmatization

```
    # Initialize the lemmatizer
    lemmatizer = WordNetLemmatizer()

    # Define a function to lemmatize the text
    def lemmatize_text(text):
        tokens = nltk.word_tokenize(text)
        lemmatized_tokens = [lemmatizer.lemmatize(token) for token in tokens]
        return ' '.join(lemmatized_tokens)

    df['question_lemmatized'] = df['question'].apply(lemmatize_text)
    df['value_lemmatized'] = df['value'].apply(lemmatize_text)
    df['round_lemmatized'] = df['round'].apply(lemmatize_text)
```

Remove punctuation and special characters

```
    df['question'] = df['question'].str.replace(r'[^\w\s]', '')
    df['value'] = df['value'].str.replace(r'[^\w\s]', '')
    df['round'] = df['round'].str.replace(r'[^\w\s]', '')
```

Remove stopwords

```
    english_stopwords = set(stopwords.words('english') + list('punct

    stop_words = english_stopwords
    stop_words.add('unknown')  # Add 'unknown' to the stopwords

    # Define a function to remove stopwords
    def remove_stopwords(text):
        tokens = nltk.word_tokenize(text)
        tokens = [token.lower() for token in tokens if token.lower()
        return ' '.join(tokens)

    # Apply the function to the question column
    df['clean_question'] = df['question'].apply(remove_stopwords)
    df['clean_value'] = df['value'].apply(remove_stopwords)
    df['clean_round'] = df['round'].apply(remove_stopwords)
```

Sarah Jackson
11:24 AM Today
(edited 11:31 AM Today)

From here on down, AI helped with extracting features, splitting, converting, and training. I could use a lot of the video and online help to cobble along the steps to prepare the data, but until AI helped show me what this might look like, I couldn't get any type of error-free code to execute anything. AI didn't give me perfect code, and I did a lot of filling in my own content, but it did help in this section, especially the code to rank the dollar values and the rounds and then print that in a data frame with a number. I was able to ask Gemini and AI by Meta help for this.

4. Extracting Features

Creating a dictionary to correlate value levels to difficulty levels import re

```
    import re

    # Normalize 'value' to an integer (e.g. "$200" -> 200, "unknown"
    def normalize_value(v):
        if v is None:
            return None
        s = str(v)
        digits = re.sub(r'\D', '', s)   # remove non-digits
        return int(digits) if digits else None

    # Map integer dollar values to difficulty levels (use integers a
    dollar_value_map = {
        200: 1,
        400: 2,
        600: 3,
        800: 4,
        1000: 5,
        1200: 6,
        1600: 7,
        2000: 8,
    }

    # Map cleaned numeric values to 1-9 difficulty levels (fallback
    df['DiffLevelVal'] = df['value'].apply(normalize_value).map(doll
```

Create a dictionary to map rounds to higher and lower difficulty levels

```
    round_type_map = {
        'jeopardy!': 1,
        'double jeopardy!': 2,
        'final jeopardy!': 3
    }

    df['DiffLevelRound'] = df['round'].map(round_type_map).fillna(9)
```

Add the two columns together

```
    # Create a new column that combines the value and round type
    df['combined_value'] = df['DiffLevelVal'] + (df['DiffLevelRound'
```

5. Splitting the Data

```
    # Split the data into 80% training and 20% testing sets
    X_train, X_test, y_train, y_test = train_test_split(df['DiffLeve
```

5. Convert to numerical format use TF-IDF to extract features

```
    # choose a text column as features and the numeric difficulty as
    X = df['clean_question']            # or 'question_lemmatized' /
    y = df['DiffLevelVal']              # or DiffLevelRound if that'

    # ensure no None values
    X = X.fillna('')
    y = y.fillna(9).astype(int)

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_s

    vectorizer = TfidfVectorizer()
    X_train_vectors = vectorizer.fit_transform(X_train.astype(str))
    X_test_vectors  = vectorizer.transform(X_test.astype(str))
```

6. Use MultinomialNB to training the model with the training data

```
    # Train a Multinomial Naive Bayesian classifier
    clf = MultinomialNB()
```

```
clf.fit(X_train_vectors, y_train)
```

```
▾ MultinomialNB
MultinomialNB()
```

11:31 AM Today

I am getting good at saving a df to a CSV! High-five me!

7. Evaluating the model

```
# Evaluate the model
y_pred = clf.predict(X_test_vectors)
print("Accuracy:", clf.score(X_test_vectors, y_test))
```

```
Accuracy: 0.19517816807265018
```

8. Wrapping Up and Saving ~ Maybe not needed but makes me feel good

```
# Save DataFrame to CSV
df.to_csv('data.csv', index=False)
# Create a DataFrame with the predictions and actual labels
predictions_df = pd.DataFrame({
    'Actual': y_test,
    'Predicted': y_pred
})

# Save the DataFrame to a CSV file
predictions_df.to_csv('predictions.csv', index=False)
```