Final Proj

May 8, 2024

How do specific game features and elements influence player satisfaction and engagement as reflected in review texts and ratings?

This question is aimed at identifying the key factors within games that contribute to or detract from player satisfaction. By analyzing the text of reviews in conjunction with the ratings, you can uncover which aspects of a game (like graphics, story, gameplay mechanics, or multiplayer features) are most closely associated with higher or lower customer ratings.

H1: Specific game features (such as story depth, graphical quality, or ease of controls) are positively correlated with higrherti

ngs. H0: Game features do not have a significant impact on the ratings.

Feature Extraction: Identify mentions of specific game features in the review texts, categorizing them into aspects like story, graphics, gameplay, sound, and multiplayer functionality. Sentiment Analysis: Perform sentiment analysis on mentions of these features to determine whether the feedback is positive, negative, or neutral. Correlation Analysis: Analyze the correlation between the presence and sentiment of these features in reviews and the overall rating of the game.

Potential Business Impact: Product Development: Insights from this analysis can guide game developers in focusing their resources on enhancing the features that players care about the most. Marketing Strategy: Marketing teams can highlight the most praised features in their campaigns to attract a larger audience based on empirical data. Customer Satisfaction: By addressing the commonly criticized aspects, developers can improve player satisfaction and potentially increase positive word-of-mouth and ratings.

[9]:
```
import nltk
from nltk import FreqDist
nltk.download("stopwords")
import pandas as pd
pd.set_option("display.max_colwidth",200)
import numpy as np
import re
import spacy
import matplotlib.pyplot as plt
import seaborn as sns
import json
import pandas as pd
import nltk
from nltk.corpus import stopwords
```

```python
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer  # Import the stemmer
from tqdm import tqdm
%matplotlib inline
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\ghlas\AppData\Roaming\nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
```

[3]:
```python
# Path to your JSON file
file_path = "S:\Python_Proj\Video_Games_5.json"

# Function to load and clean data
def load_and_clean_data(file_path):
    data = []
    with open(file_path, 'r', encoding='utf-8') as file:
        for line in file:
            try:
                # Load line as JSON and append to list
                entry = json.loads(line)
                # Remove unwanted columns
                entry.pop('unixReviewTime', None)
                entry.pop('vote', None)
                entry.pop('style', None)
                entry.pop('image', None)
                entry.pop('reviewTime', None)
                entry.pop('reviewerName', None)
                entry.pop('verified', None)
                entry.pop('summary', None)
                data.append(entry)
            except json.JSONDecodeError:
                continue
    return pd.DataFrame(data)

# Load the dataset and clean it
df = load_and_clean_data(file_path)
print(df.head())
```

```
<>:2: SyntaxWarning: invalid escape sequence '\P'
<>:2: SyntaxWarning: invalid escape sequence '\P'
C:\Users\ghlas\AppData\Local\Temp\ipykernel_31372\2746396954.py:2:
SyntaxWarning: invalid escape sequence '\P'
  file_path = "S:\Python_Proj\Video_Games_5.json"
```

```
   overall      reviewerID        asin  \
0      5.0  A1HP7NVNPFMA4N  0700026657
1      4.0  A1JGAP0185YJI6  0700026657
2      3.0  A1YJWEXHQBWK2B  0700026657
3      2.0  A2204E1TH211HT  0700026657
```

```
4      5.0  A2RF5B5H74JLPE  0700026657

                                     reviewText
0
This game is a bit hard to get the hang of, but when you do it's great.
1  I played it a while but it was alright. The steam was a bit of trouble. The
more they move these game to steam the more of a hard time I have activating and
playing a game. But in spite of that it…
2
ok game.
3
found the game a bit too complicated, not what I expected after having played
1602, 1503, and 1701
4
great game, I love it and have played it since its arrived
```

[4]:
```python
import pandas as pd

# Drop rows where 'reviewText' or 'overall' might be missing
df.dropna(subset=['reviewText', 'overall'], inplace=True)

# Explicitly convert review texts to string to avoid any confusion
df['reviewText'] = df['reviewText'].astype(str)
df['reviewText'] = df['reviewText'].str.replace(r"[^a-zA-Z\s]", "", regex=True).
 ↪str.lower()

# Create a new column to store the length of each review
df['review_length'] = df['reviewText'].apply(len)
all_strings = all(isinstance(x, str) for x in df['reviewText'])
print("All entries are strings:", all_strings)

# Display basic information about the DataFrame to ensure cleanliness
print(df.info())

# Display basic statistics of the review lengths
print(df['review_length'].describe())

print("Reviews longer than 2000 characters:", (df['review_length'] >= 1000).
 ↪sum())
df = df[df['review_length'] >= 1000]

# Print updated DataFrame info to confirm changes
print(df.info())
```

```
All entries are strings: True
<class 'pandas.core.frame.DataFrame'>
Index: 497419 entries, 0 to 497576
Data columns (total 5 columns):
 #   Column         Non-Null Count   Dtype  
---  ------         --------------   -----  
 0   overall        497419 non-null  float64
 1   reviewerID     497419 non-null  object 
 2   asin           497419 non-null  object 
 3   reviewText     497419 non-null  object 
 4   review_length  497419 non-null  int64  
dtypes: float64(1), int64(1), object(3)
memory usage: 22.8+ MB
None
count    497419.00000
mean        646.58007
std        1221.11558
min           0.00000
25%          55.00000
50%         203.00000
75%         685.00000
max       31885.00000
Name: review_length, dtype: float64
Reviews longer than 2000 characters: 90182
<class 'pandas.core.frame.DataFrame'>
Index: 90182 entries, 13 to 497575
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   overall        90182 non-null  float64
 1   reviewerID     90182 non-null  object 
 2   asin           90182 non-null  object 
 3   reviewText     90182 non-null  object 
 4   review_length  90182 non-null  int64  
dtypes: float64(1), int64(1), object(3)
memory usage: 4.1+ MB
None
```

[5]:
```python
# Create a new column to store the length of each review
df['review_length'] = df['reviewText'].apply(len)
print(df['review_length'].describe())
```

```
count    90182.000000
mean      2512.675722
std       1921.096473
min       1000.000000
25%       1331.000000
50%       1871.000000
75%       2971.000000
max      31885.000000
Name: review_length, dtype: float64
```

[6]:
```
nltk.download('punkt')
nltk.download('stopwords')

# Load spaCy's English language model
nlp = spacy.load('en_core_web_sm', disable=['parser', 'ner'])

# Initialize the Porter stemmer (from previous code)
from nltk.stem import PorterStemmer
stemmer = PorterStemmer()

# Function to preprocess text using both NLTK and spaCy for lemmatization
def preprocess_text(text):
    # Convert text to lowercase
    text = text.lower()
    # Tokenize text using NLTK
    tokens = word_tokenize(text)
    # Remove stop words using NLTK
    stop_words = set(stopwords.words('english'))
    filtered_tokens = [word for word in tokens if word not in stop_words and␣
 ↪word.isalpha()]

    # Create a spaCy document for lemmatization
    doc = nlp(' '.join(filtered_tokens))
    lemmatized = [token.lemma_ for token in doc]

    return ' '.join(lemmatized)

#  tqdm
tqdm.pandas()

df['processed_reviewText'] = df['reviewText'].progress_apply(preprocess_text)
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\ghlas\AppData\Roaming\nltk_data…
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\ghlas\AppData\Roaming\nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
100%|
| 90182/90182 [13:46<00:00, 109.17it/s]
```

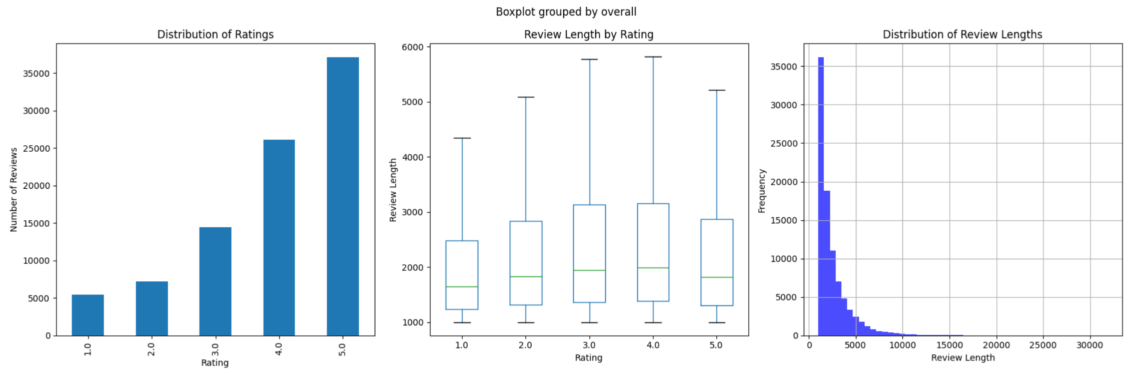
[7]:
```
# Display the first few processed texts
df = df.dropna(subset=['processed_reviewText'])
df = df[df['processed_reviewText'] != "nan"]
print(df[['reviewText', 'processed_reviewText']].head())
```

```
                                             reviewText  \
13  im not quite finished with the games dirt tour mode but i believe ive
```

```
experienced the bulk of what the game has to offer  and im happy to say that the
game is indeed awesome  great cars great trac…
16  this is a must have for any gamer codemasters really hit a home run on this
one i would like to see f  be this good using the ego engine  and the physics
model in this game is right on the money t…
18  update june \ndeeply disappointed at the lack of a nonlinear openworld
environment but its the restrictive qte gameplay and finalkill cutscenes that
kill the game for me i loath enforced qte when …
19  i will open with the pros\nreplayability  its hitman what did you expect you
can go back and replay missions over and over to try for a cleaner quicker kill
or just fly right through it in a weeke…
29  dirt  was like this  im becoming more  more suspicious of games like this
that require a continuous open internet to play  it just seems like bait set out
for some young naive prey to fall for  do…

                       processed_reviewText
13  I m quite finished game dirt tour mode believe I ve experience bulk game
offer I m happy say game indeed awesome great car great track race mode
excellent gameplay graphic highlight race snow vari…
16  must gamer codemaster really hit home run one would like see f good use ego
engine physics model game right money game reason everyone buy good graphic card
afford use nvidia overclocked order wan…
18  update june deeply disappointed lack nonlinear openworld environment
restrictive qte gameplay finalkill cutscene kill game loath enforce qte mission
go awry you re try escape hail bullet happen br…
19  open pro replayability hitman expect go back replay mission try clean quick
kill fly right weekend like length game listen reviews xbox official magazine
say lengthy game finish one weekend friday…
29  dirt like I m become suspicious game like require continuous open internet
play seem like bait set young naive prey fall anyone honestly believe personal
information collect particular reason do n…
```

[11]:
```python
import matplotlib.pyplot as plt

fig, axs = plt.subplots(1, 3, figsize=(18, 6))

# Plot 1: Distribution of Ratings
df['overall'].value_counts().sort_index().plot(kind='bar', ax=axs[0])
axs[0].set_title('Distribution of Ratings')
axs[0].set_xlabel('Rating')
axs[0].set_ylabel('Number of Reviews')

# Plot 2: Review Length by Rating
df['review_length'] = df['reviewText'].apply(len)
df.boxplot(column='review_length', by='overall', grid=False, showfliers=False,␣
 ↪ax=axs[1])
axs[1].set_title('Review Length by Rating')
```

```
axs[1].set_xlabel('Rating')
axs[1].set_ylabel('Review Length')

# Plot 3: Distribution of Review Lengths
axs[2].hist(df['review_length'], bins=50, color='blue', alpha=0.7)
axs[2].set_title('Distribution of Review Lengths')
axs[2].set_xlabel('Review Length')
axs[2].set_ylabel('Frequency')
axs[2].grid(True)

fig.tight_layout()
plt.show()
```



[12]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.decomposition import TruncatedSVD  # Import TruncatedSVD for LSA
from tqdm.notebook import tqdm  #   tqdm   tqdm_notebook

import pandas as pd

#
tfidf_vectorizer = TfidfVectorizer(max_features=4000, stop_words='english')

#
dtm = tfidf_vectorizer.fit_transform(df['processed_reviewText'])

#
n_topics = 5

#  LSA
lsa = TruncatedSVD(n_components=n_topics, random_state=42)

#   tqdm
tqdm.pandas()
```

```
#  LSA
lsa.fit(dtm)

#
feature_names = tfidf_vectorizer.get_feature_names_out()
for topic_idx, topic in enumerate(tqdm(lsa.components_, desc='LSA Topic␣
 ↪Progress')):
    print(f"Topic {topic_idx}:")
    top_words_idx = topic.argsort()[:-11:-1]
    top_words = [feature_names[i] for i in top_words_idx]
    weights = [topic[i] for i in top_words_idx]
    print("\n".join([f"{word}: {weight}" for word, weight in zip(top_words,␣
 ↪weights)]))
```

```
LSA Topic Progress:   0%|          | 0/5 [00:00<?, ?it/s]

Topic 0:
game: 0.5236244122059027
play: 0.18016300135389915
like: 0.15390090028174647
time: 0.12125106375371393
good: 0.11774664023143717
character: 0.11651317305832855
make: 0.11146512667471252
really: 0.10702334038358467
story: 0.09953572496652699
fun: 0.09458125696402175
Topic 1:
controller: 0.37072029102441384
mouse: 0.33712381880502046
headset: 0.22997047448555222
button: 0.20074695751350788
use: 0.1702243690790324
ps: 0.16022573870845844
xbox: 0.15942834844076498
keyboard: 0.1370208459825826
work: 0.1167038592139729
wii: 0.11474022584685982
Topic 2:
mouse: 0.4852729581273917
enemy: 0.13780490998113784
character: 0.13449249645141031
keyboard: 0.12804342628630733
button: 0.11370904064892126
weapon: 0.1103817621713392
story: 0.10149601931635355
mission: 0.09549248470111553
```

```
key: 0.09542244075271651
battle: 0.08306624491810805
Topic 3:
mouse: 0.5003042692820453
mario: 0.29578819879605706
wii: 0.22715402697831813
game: 0.1505154629003359
nintendo: 0.12958982627848972
button: 0.11184146424872854
ds: 0.09865645348275648
super: 0.09307656643642526
kart: 0.0778941735869288
dpi: 0.07190828996205774
Topic 4:
car: 0.41408350559175955
race: 0.25864321152418523
sim: 0.18518189085723552
drive: 0.13558747056781076
mouse: 0.13182663469719985
mission: 0.11715130360674132
racing: 0.10110027212566493
online: 0.09427407254448114
mode: 0.08656292331903907
track: 0.08509581399701571
```

[13]:
```
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.decomposition import TruncatedSVD
from sklearn.preprocessing import normalize
from tqdm.notebook import tqdm

# Define informative topics and their keywords
informative_topics = {
    'Graphics': [
        'resolution', 'textures', 'lighting', 'shadows', 'modeling',␣
↪'raytracing', 'HDR', 'framerate',
        'aliasing', 'rendering', '3D', 'detail', 'visuals', 'graphics',␣
↪'polygons', 'animation',
        'realism', 'artistic', 'effects', 'visual quality', 'color depth',␣
↪'bloom'
    ],
    'Story': [
        'plot', 'narrative', 'characters', 'dialogue', 'lore', 'cutscenes',␣
↪'script', 'storytelling',
        'development', 'twists', 'narration', 'themes', 'engagement', 'story␣
↪arcs', 'character arcs',
        'motivation', 'backstory', 'prologue', 'epilogue', 'drama', 'conflict',␣
 ↪'resolution'
    ],
    'Gameplay': [
        'mechanics', 'controls', 'design', 'difficulty', 'tutorials',␣
 ↪'interaction', 'AI', 'strategy',
        'levels', 'campaign', 'missions', 'challenges', 'playability', 'user␣
 ↪interface', 'HUD',
        'customization', 'skills', 'abilities', 'game balance', 'level␣
 ↪progression', 'sandbox',
        'open world', 'quest design'
    ],
    'Sound': [
        'soundtrack', 'effects', 'voices', 'audio', 'ambient', 'fidelity',␣
 ↪'acoustics', 'clarity',
        'mixing', '3Dsound', 'volume', 'music', 'sound design', 'dialogue␣
 ↪clarity', 'audio immersion',
        'surround sound', 'soundscapes', 'score', 'audio cues', 'background␣
 ↪noise', 'voiceovers'
    ],
    'Multiplayer': [
        'servers', 'matchmaking', 'co-op', 'pvp', 'esports', 'community',␣
 ↪'online', 'multiplayer',
        'teamplay', 'network', 'lobby', 'interaction', 'connectivity', 'social␣
 ↪features', 'clans',
        'guilds', 'teams', 'voice chat', 'competitive', 'leaderboards',␣
 ↪'rankings', 'cross-platform play'
    ]
}
```

[14]:
```
text_data = df['processed_reviewText']
# Vectorize the text data
vectorizer = CountVectorizer(max_features=1000, stop_words='english')
dtm = vectorizer.fit_transform(text_data)
# Apply Latent Semantic Analysis (LSA)
n_topics = 5
lsa = TruncatedSVD(n_components=n_topics, random_state=42)
lsa.fit(dtm)
dtm_lsa = lsa.transform(dtm)
# Normalize the LSA results
dtm_lsa_normalized = normalize(dtm_lsa, axis=1)
# Create a DataFrame to store LSA topic weights for each review
df_lsa_topics = pd.DataFrame(dtm_lsa_normalized, columns=[f'LSA_Topic_{i}' for␣
 ↪i in range(n_topics)])
```

[15]:
```
import pandas as pd
from tqdm.auto import tqdm
# DataFrame to store topic assignments based on keyword presence
df_informative_topics = pd.DataFrame(index=df.index)

# Iterate over each topic and check if any of its keywords are in the␣
 ↪processed_reviewText
for topic, keywords in tqdm(informative_topics.items(), desc="Processing␣
 ↪topics"):
    # Create a regex pattern that matches any of the keywords
    pattern = '|'.join([f'\\b{keyword}\\b' for keyword in keywords])  # \b is␣
 ↪used to ensure full words match
    df_informative_topics[topic] = df['processed_reviewText'].str.
 ↪contains(pattern, case=False, na=False)

print(df_informative_topics.head())
```

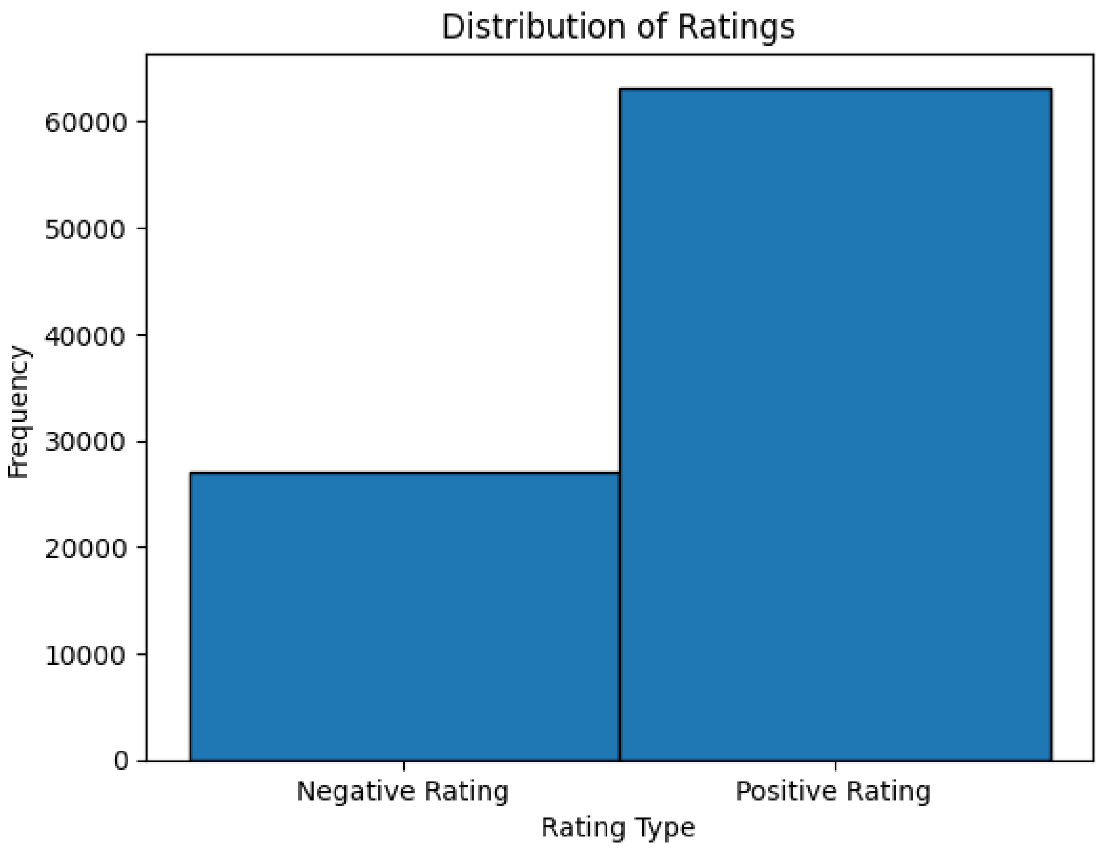
```
Processing topics:   0%|          | 0/5 [00:00<?, ?it/s]

    Graphics  Story  Gameplay  Sound  Multiplayer
13     False  False      True  False        False
16      True  False     False  False        False
18     False  False      True  False         True
19     False   True     False   True        False
29     False  False      True   True         True
```

[16]:
```
# Concatenate this DataFrame with the original DataFrame (or with LSA topics␣
 ↪DataFrame)
df_with_all_topics = pd.concat([df, df_lsa_topics, df_informative_topics],␣
 ↪axis=1)
df_with_all_topics = df_with_all_topics.dropna(subset=['processed_reviewText'])
df_with_all_topics =␣
 ↪df_with_all_topics[df_with_all_topics['processed_reviewText'] != "nan"]
print(df_with_all_topics[['reviewText', 'processed_reviewText']].head())
```

```
                                          reviewText  \
13  im not quite finished with the games dirt tour mode but i believe ive
experienced the bulk of what the game has to offer  and im happy to say that the
game is indeed awesome  great cars great trac…
16  this is a must have for any gamer codemasters really hit a home run on this
one i would like to see f  be this good using the ego engine  and the physics
model in this game is right on the money t…
18  update june \ndeeply disappointed at the lack of a nonlinear openworld
environment but its the restrictive qte gameplay and finalkill cutscenes that
kill the game for me i loath enforced qte when …
19  i will open with the pros\nreplayability  its hitman what did you expect you
can go back and replay missions over and over to try for a cleaner quicker kill
or just fly right through it in a weeke…
```

```
29  dirt  was like this  im becoming more  more suspicious of games like this
that require a continuous open internet to play  it just seems like bait set out
for some young naive prey to fall for  do…

                       processed_reviewText
13  I m quite finished game dirt tour mode believe I ve experience bulk game
offer I m happy say game indeed awesome great car great track race mode
excellent gameplay graphic highlight race snow vari…
16  must gamer codemaster really hit home run one would like see f good use ego
engine physics model game right money game reason everyone buy good graphic card
afford use nvidia overclocked order wan…
18  update june deeply disappointed lack nonlinear openworld environment
restrictive qte gameplay finalkill cutscene kill game loath enforce qte mission
go awry you re try escape hail bullet happen br…
19  open pro replayability hitman expect go back replay mission try clean quick
kill fly right weekend like length game listen reviews xbox official magazine
say lengthy game finish one weekend friday…
29  dirt like I m become suspicious game like require continuous open internet
play seem like bait set young naive prey fall anyone honestly believe personal
information collect particular reason do n…
```

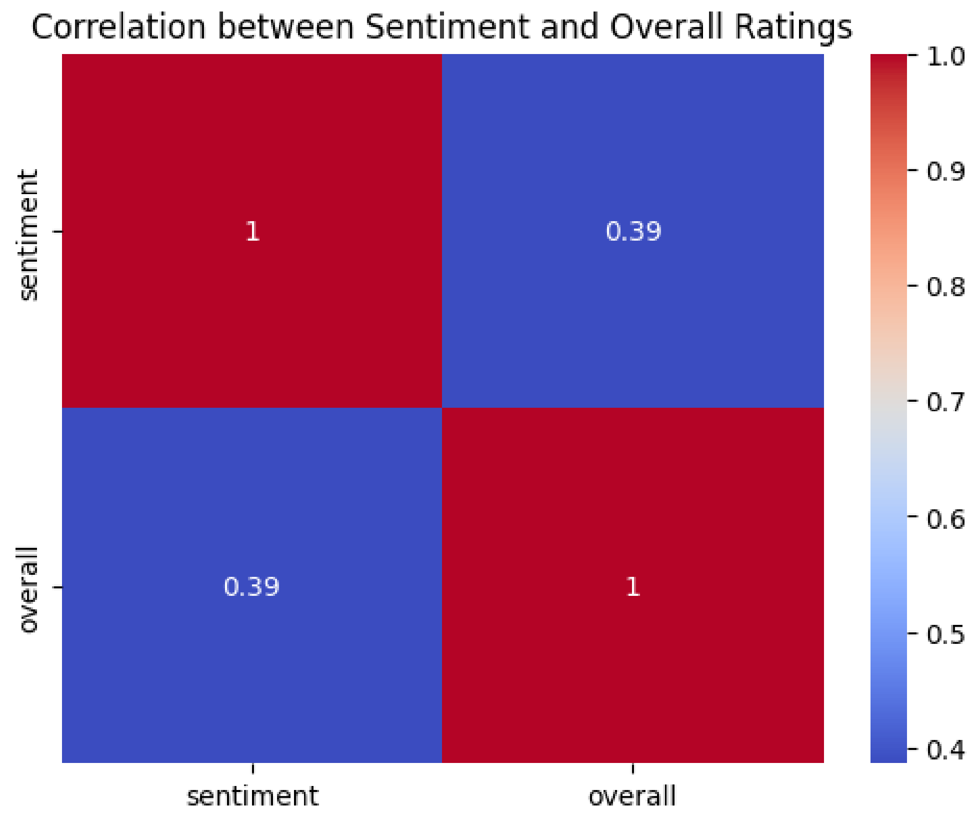
[17]:
```python
import numpy as np
import matplotlib.pyplot as plt

df_with_all_topics['overall_binary'] = np.where(df_with_all_topics['overall']␣
 ↪>= 4, 1, 0)

# Plotting the distribution of the new binary variable
plt.hist(df_with_all_topics['overall_binary'], bins=[-0.5, 0.5, 1.5],␣
 ↪edgecolor='black')
plt.xticks([0, 1], ['Negative Rating', 'Positive Rating'])
plt.xlabel('Rating Type')
plt.ylabel('Frequency')
plt.title('Distribution of Ratings')
plt.show()
```



[18]:
```
from textblob import TextBlob

df_with_all_topics['processed_reviewText'] =␣
↪df_with_all_topics['processed_reviewText'].astype(str)  # Ensure all data is␣
↪treated as string

# Function to compute sentiment from text
def compute_sentiment(text):
    if text.strip():
        return TextBlob(text).sentiment.polarity
    else:
        return 0

# Apply sentiment analysis
df_with_all_topics['sentiment'] = df_with_all_topics['processed_reviewText'].
↪apply(compute_sentiment)

# Display sentiment scores
print(df_with_all_topics[['processed_reviewText', 'sentiment']])
```

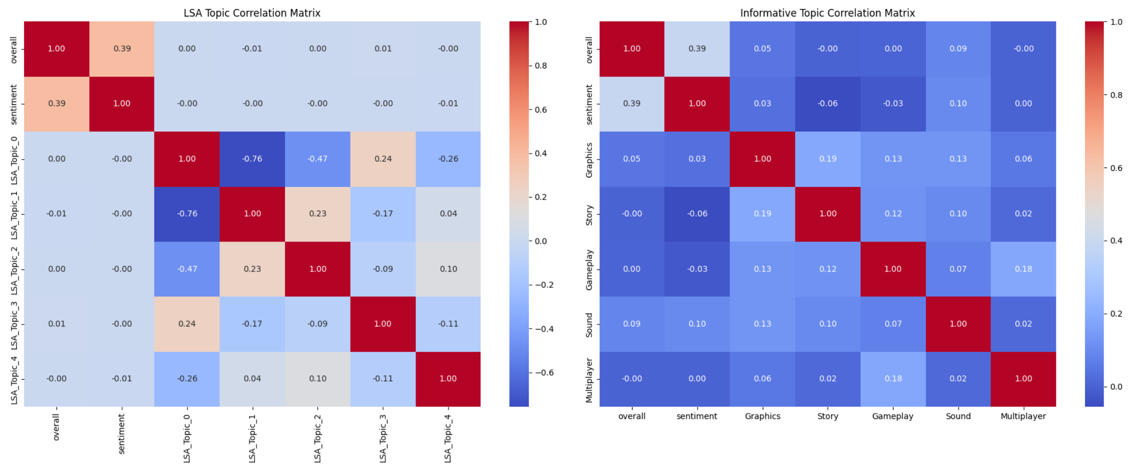
```
                           processed_reviewText  \
13      I m quite finished game dirt tour mode believe I ve experience bulk game
offer I m happy say game indeed awesome great car great track race mode
excellent gameplay graphic highlight race snow vari…
16      must gamer codemaster really hit home run one would like see f good use
ego engine physics model game right money game reason everyone buy good graphic
card afford use nvidia overclocked order wan…
18      update june deeply disappointed lack nonlinear openworld environment
restrictive qte gameplay finalkill cutscene kill game loath enforce qte mission
go awry you re try escape hail bullet happen br…
19      open pro replayability hitman expect go back replay mission try clean
quick kill fly right weekend like length game listen reviews xbox official
magazine say lengthy game finish one weekend friday…
29      dirt like I m become suspicious game like require continuous open
internet play seem like bait set young naive prey fall anyone honestly believe
personal information collect particular reason do n…
…
…
497454  want thank reviewer talk game recommend long time think cheaply make
first person parkour runner type game way wrong thinking wasent guy would never
buy look ps exclusive play since everything ste…
497462  back original dishonor release dishonor everything fan would want sequel
though small detail deep focus story make worthy successor franchise set several
year event first game dishonor pick story …
497538  buy game directly steam finish unlocked unlocked review pro definitely
scare moment except first person view definitely old feel want hoard ammo stuff
puzzle solve decent story decent visual excep…
497561  other may already state farm sim blast play busy busy sim farmer testify
level authenticity say challenge nothing else always busy something think
simfarm could reality play nonstop last four day …
497575  think originally begin play bioshock several year ago be not level
enough do not enough ammo remember reach point battle big daddy wrench fight
ammo exhausted hour game do not want go back never p…

        sentiment
13       0.167435
16       0.066181
18      -0.046354
19      -0.038515
29      -0.009066
…            …
497454   0.095455
497462   0.109076
497538   0.046552
497561   0.081775
497575   0.015066

[90182 rows x 2 columns]
```

[19]:
```python
import seaborn as sns

# Calculate correlations between sentiment scores and overall ratings
correlations = df_with_all_topics[['sentiment', 'overall']].corr()

# Visualize the correlation matrix
sns.heatmap(correlations, annot=True, cmap='coolwarm')
plt.title('Correlation between Sentiment and Overall Ratings')
plt.show()
```



[32]:
```python
import seaborn as sns
import matplotlib.pyplot as plt

lsa_topics = ['LSA_Topic_0', 'LSA_Topic_1', 'LSA_Topic_2', 'LSA_Topic_3',␣
 ↪'LSA_Topic_4']
informative_topic_names = list(informative_topics.keys())

# Compute correlation matrices
lsa_correlation_matrix = df_with_all_topics[['overall', 'sentiment'] +␣
 ↪lsa_topics].corr()
```

```
informative_correlation_matrix = df_with_all_topics[['overall', 'sentiment'] +␣
 ↪informative_topic_names].corr()

# Create subplots
fig, axs = plt.subplots(1, 2, figsize=(20, 8))  # 1 row, 2 columns

# Plot LSA Topic Correlation Matrix
sns.heatmap(lsa_correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f",␣
 ↪ax=axs[0])
axs[0].set_title('LSA Topic Correlation Matrix')

# Plot Informative Topic Correlation Matrix
sns.heatmap(informative_correlation_matrix, annot=True, cmap='coolwarm', fmt=".
 ↪2f", ax=axs[1])
axs[1].set_title('Informative Topic Correlation Matrix')

# Adjust layout for better spacing
fig.tight_layout()
plt.show()
```



```
[21]: import numpy as np
      import pandas as pd
      from sklearn.model_selection import train_test_split, KFold, cross_val_score
      from sklearn.ensemble import RandomForestClassifier
      from sklearn.neural_network import MLPClassifier
      from sklearn.impute import SimpleImputer
      from sklearn.pipeline import make_pipeline
      from sklearn.metrics import accuracy_score, classification_report
      import xgboost as xgb
      from sklearn.metrics import accuracy_score
```

```
# Split the dataset
train, test = train_test_split(df_with_all_topics, test_size=0.2,␣
↪random_state=42)

# LSA topics
X_train_lsa = train[['LSA_Topic_0', 'LSA_Topic_1', 'LSA_Topic_2',␣
↪'LSA_Topic_3', 'LSA_Topic_4']]
y_train_lsa = train['overall_binary']
X_test_lsa = test[['LSA_Topic_0', 'LSA_Topic_1', 'LSA_Topic_2', 'LSA_Topic_3',␣
↪'LSA_Topic_4']]
y_test_lsa = test['overall_binary']

# Informative topics
informative_topic_names = list(informative_topics.keys())
X_train_info = train[informative_topic_names]
y_train_info = train['overall_binary']
X_test_info = test[informative_topic_names]
y_test_info = test['overall_binary']

def evaluate_model(model, X_train, y_train, X_test, y_test):
    model.fit(X_train, y_train)
    y_pred_train = model.predict(X_train)
    y_pred_test = model.predict(X_test)
    accuracy_train = accuracy_score(y_train, y_pred_train)
    accuracy_test = accuracy_score(y_test, y_pred_test)
    print(classification_report(y_test, y_pred_test))
    return accuracy_train, accuracy_test

def evaluate_model_xgb(params, dtrain, dtest, num_rounds=100):
    # Train the model
    bst = xgb.train(params, dtrain, num_boost_round=num_rounds)

    # Make predictions
    y_pred_train = (bst.predict(dtrain) > 0.5).astype(int)
    y_pred_test = (bst.predict(dtest) > 0.5).astype(int)

    # Calculate accuracy
    accuracy_train = accuracy_score(dtrain.get_label(), y_pred_train)
    accuracy_test = accuracy_score(dtest.get_label(), y_pred_test)

    return accuracy_train, accuracy_test
```

```
[22]: import pandas as pd
import statsmodels.api as sm
import statsmodels.formula.api as smf
# Formula for LSA Topics
```

```python
formula_lsa = 'overall_binary ~ LSA_Topic_0 + LSA_Topic_1 + LSA_Topic_2 +␣
 ↪LSA_Topic_3 + LSA_Topic_4'

# Fitting the OLS model
model_lsa = smf.ols(formula=formula_lsa, data=df_with_all_topics)
results_lsa = model_lsa.fit()
print("OLS Results for LSA Topics:")
print(results_lsa.summary())

# Formula for Informative Topics
formula_info = 'overall_binary ~ Graphics + Story + Gameplay + Sound +␣
 ↪Multiplayer'

# Fitting the OLS model
model_info = smf.ols(formula=formula_info, data=df_with_all_topics)
results_info = model_info.fit()
print("OLS Results for Informative Topics:")
print(results_info.summary())
# Logistic Regression for LSA Topics
model_lsa_logit = smf.logit(formula=formula_lsa, data=df_with_all_topics)
results_lsa_logit = model_lsa_logit.fit()
print("Logistic Regression Results for LSA Topics:")
print(results_lsa_logit.summary())

# Logistic Regression for Informative Topics
model_info_logit = smf.logit(formula=formula_info, data=df_with_all_topics)
results_info_logit = model_info_logit.fit()
print("Logistic Regression Results for Informative Topics:")
print(results_info_logit.summary())
```

```
OLS Results for LSA Topics:
                            OLS Regression Results
==============================================================================
Dep. Variable:         overall_binary   R-squared:                       0.000
Model:                            OLS   Adj. R-squared:                 -0.000
Method:                 Least Squares   F-statistic:                    0.6253
Date:                Wed, 08 May 2024   Prob (F-statistic):              0.680
Time:                        20:54:22   Log-Likelihood:                -15894.
No. Observations:               27475   AIC:                         3.180e+04
Df Residuals:                   27469   BIC:                         3.185e+04
Df Model:                           5
Covariance Type:            nonrobust
===============================================================================
                  coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------
Intercept       0.7974      0.032     24.839      0.000       0.735       0.860
LSA_Topic_0    -0.0491      0.035     -1.415      0.157      -0.117       0.019
```

```
LSA_Topic_1    -0.0189      0.015     -1.253      0.210      -0.049       0.011
LSA_Topic_2     0.0001      0.015      0.010      0.992      -0.030       0.030
LSA_Topic_3     0.0115      0.014      0.819      0.413      -0.016       0.039
LSA_Topic_4    -0.0142      0.017     -0.836      0.403      -0.047       0.019
==============================================================================
Omnibus:                     5123.426   Durbin-Watson:                   1.779
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             6725.031
Skew:                          -1.170   Prob(JB):                         0.00
Kurtosis:                       2.370   Cond. No.                         25.6
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
OLS Results for Informative Topics:
                            OLS Regression Results
==============================================================================
Dep. Variable:          overall_binary   R-squared:                       0.008
Model:                             OLS   Adj. R-squared:                  0.008
Method:                  Least Squares   F-statistic:                     147.8
Date:                 Wed, 08 May 2024   Prob (F-statistic):          7.27e-157
Time:                         20:54:22   Log-Likelihood:                -57171.
No. Observations:                90182   AIC:                         1.144e+05
Df Residuals:                    90176   BIC:                         1.144e+05
Df Model:                            5
Covariance Type:             nonrobust
================================================================================
=======
                          coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
-------
Intercept               0.6745      0.002    280.321      0.000       0.670
0.679
Graphics[T.True]        0.0403      0.004     10.661      0.000       0.033
0.048
Story[T.True]          -0.0201      0.004     -5.160      0.000      -0.028
-0.012
Gameplay[T.True]       -0.0050      0.003     -1.565      0.118      -0.011
0.001
Sound[T.True]           0.0801      0.003     23.672      0.000       0.073
0.087
Multiplayer[T.True]     0.0002      0.003      0.052      0.959      -0.007
0.007
==============================================================================
Omnibus:                   102055.469   Durbin-Watson:                   1.756
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            16774.887
Skew:                          -0.864   Prob(JB):                         0.00
```

```
Kurtosis:                       1.784   Cond. No.                         3.39
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
Optimization terminated successfully.
         Current function value: 0.559472
         Iterations 5
Logistic Regression Results for LSA Topics:
                           Logit Regression Results
==============================================================================
Dep. Variable:         overall_binary   No. Observations:                27475
Model:                          Logit   Df Residuals:                    27469
Method:                           MLE   Df Model:                            5
Date:                Wed, 08 May 2024   Pseudo R-squ.:               0.0001020
Time:                        20:54:22   Log-Likelihood:                -15371.
converged:                       True   LL-Null:                       -15373.
Covariance Type:            nonrobust   LLR p-value:                    0.6790
===============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
-------------------------------------------------------------------------------
Intercept      1.3545      0.173      7.823      0.000       1.015       1.694
LSA_Topic_0   -0.2646      0.187     -1.416      0.157      -0.631       0.102
LSA_Topic_1   -0.1019      0.081     -1.254      0.210      -0.261       0.057
LSA_Topic_2    0.0008      0.082      0.009      0.993      -0.160       0.161
LSA_Topic_3    0.0622      0.076      0.821      0.412      -0.086       0.211
LSA_Topic_4   -0.0762      0.091     -0.837      0.402      -0.254       0.102
===============================================================================
Optimization terminated successfully.
         Current function value: 0.606178
         Iterations 5
Logistic Regression Results for Informative Topics:
                           Logit Regression Results
==============================================================================
Dep. Variable:         overall_binary   No. Observations:                90182
Model:                          Logit   Df Residuals:                    90176
Method:                           MLE   Df Model:                            5
Date:                Wed, 08 May 2024   Pseudo R-squ.:                0.006836
Time:                        20:54:23   Log-Likelihood:                -54666.
converged:                       True   LL-Null:                       -55043.
Covariance Type:            nonrobust   LLR p-value:                 2.117e-160
================================================================================
=======
                         coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------------
-------
```

```
Intercept               0.7279      0.011     63.757      0.000       0.706
0.750
Graphics[T.True]        0.1993      0.019     10.689      0.000       0.163
0.236
Story[T.True]          -0.0969      0.019     -5.193      0.000      -0.134
-0.060
Gameplay[T.True]       -0.0235      0.015     -1.546      0.122      -0.053
0.006
Sound[T.True]           0.3984      0.017     23.555      0.000       0.365
0.432
Multiplayer[T.True]     0.0010      0.017      0.059      0.953      -0.032
0.034
================================================================================
=======
```

[23]:

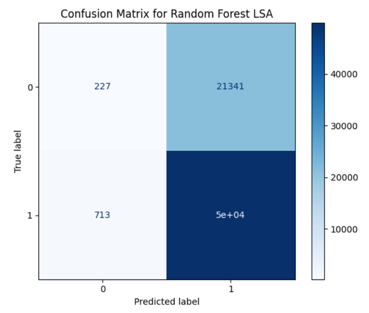
```python
from sklearn.model_selection import cross_val_score

# Random Forest - LSA
rf_lsa = RandomForestClassifier(n_estimators=50, random_state=42)
# Applying 5-fold Cross-Validation
cv_scores_lsa = cross_val_score(rf_lsa, X_train_lsa, y_train_lsa, cv=5,␣
 ↪scoring='accuracy')

print("CV Scores for Random Forest LSA:", cv_scores_lsa)
print("Mean CV Accuracy for Random Forest LSA:", cv_scores_lsa.mean())

print("\n")


rf_info = RandomForestClassifier(n_estimators=50, random_state=42)
# Applying 5-fold Cross-Validation
cv_scores_info = cross_val_score(rf_info, X_train_info, y_train_info, cv=5,␣
 ↪scoring='accuracy')
# Print the CV scores
print("CV Scores for Random Forest Informative Topics:", cv_scores_info)
print("Mean CV Accuracy for Random Forest Informative Topics:", cv_scores_info.
 ↪mean())
```

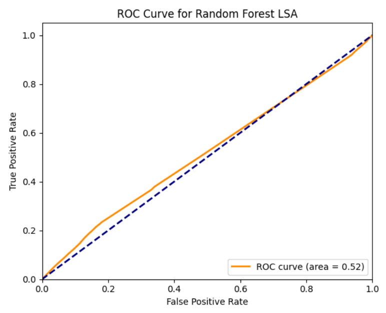
```
CV Scores for Random Forest LSA: [0.6927715  0.69644466 0.69575161 0.69443482
0.69297942]
Mean CV Accuracy for Random Forest LSA: 0.694476401691039


CV Scores for Random Forest Informative Topics: [0.70108809 0.70108809
0.70101878 0.70101878 0.70101878]
Mean CV Accuracy for Random Forest Informative Topics: 0.701046503569201
```

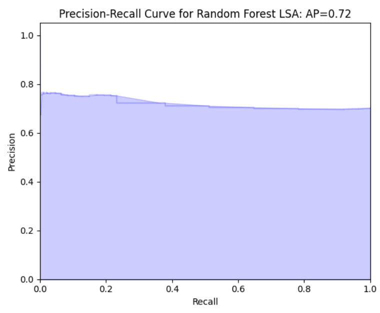
```
[24]: import numpy as np
      import matplotlib.pyplot as plt
      from sklearn.model_selection import KFold
      from sklearn.ensemble import RandomForestClassifier
      from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay,␣
       ↪roc_curve, auc, precision_recall_curve, average_precision_score,␣
       ↪accuracy_score

      # Function to perform CV and collect predictions
      def cross_val_predict_proba(model, X, y, n_splits=5):
          kf = KFold(n_splits=n_splits, shuffle=True, random_state=42)
          y_trues, y_preds, y_scores = [], [], []

          for train_index, test_index in kf.split(X):
              X_train, X_test = X.iloc[train_index], X.iloc[test_index]
              y_train, y_test = y.iloc[train_index], y.iloc[test_index]

              model.fit(X_train, y_train)
              # Collect predictions
              y_pred = model.predict(X_test)
              y_prob = model.predict_proba(X_test)[:, 1]  # assuming binary␣
       ↪classification

              y_trues.extend(y_test)
              y_preds.extend(y_pred)
              y_scores.extend(y_prob)

          return y_trues, y_preds, y_scores

      # Plotting functions
      def plot_evaluation_metrics(y_true, y_pred, y_scores, model_name):
          fig, axes = plt.subplots(nrows=1, ncols=3, figsize=(18, 5))

          # Confusion Matrix
          ax1 = axes[0]
          cm = confusion_matrix(y_true, y_pred)
          disp = ConfusionMatrixDisplay(confusion_matrix=cm)
          disp.plot(ax=ax1, cmap=plt.cm.Blues)
          ax1.set_title(f'Confusion Matrix for {model_name}')

          # ROC Curve
          ax2 = axes[1]
          fpr, tpr, _ = roc_curve(y_true, y_scores)
          roc_auc = auc(fpr, tpr)
          ax2.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area =␣
       ↪{roc_auc:.2f})')
          ax2.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
```

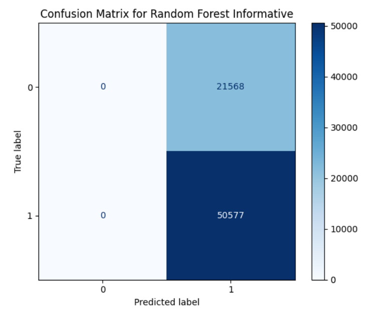
```
    ax2.set_xlim([0.0, 1.0])
    ax2.set_ylim([0.0, 1.05])
    ax2.set_xlabel('False Positive Rate')
    ax2.set_ylabel('True Positive Rate')
    ax2.set_title(f'ROC Curve for {model_name}')
    ax2.legend(loc="lower right")

    # Precision-Recall Curve
    ax3 = axes[2]
    precision, recall, _ = precision_recall_curve(y_true, y_scores)
    ap_score = average_precision_score(y_true, y_scores)
    ax3.step(recall, precision, color='b', alpha=0.2, where='post')
    ax3.fill_between(recall, precision, alpha=0.2, color='b')
    ax3.set_xlabel('Recall')
    ax3.set_ylabel('Precision')
    ax3.set_ylim([0.0, 1.05])
    ax3.set_xlim([0.0, 1.0])
    ax3.set_title(f'Precision-Recall Curve for {model_name}: AP={ap_score:.2f}')

    plt.tight_layout()
    plt.show()
```

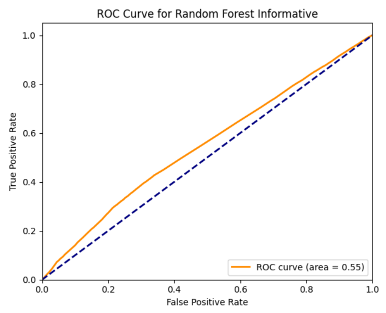
[25]:
```
# Random Forest - LSA
rf_lsa = RandomForestClassifier(n_estimators=50, random_state=42)
y_true_lsa, y_pred_lsa, y_scores_lsa = cross_val_predict_proba(rf_lsa,␣
 ↪X_train_lsa, y_train_lsa)

# Random Forest - Informative Topics
rf_info = RandomForestClassifier(n_estimators=50, random_state=42)
y_true_info, y_pred_info, y_scores_info = cross_val_predict_proba(rf_info,␣
 ↪X_train_info, y_train_info)

# Plot metrics for LSA
plot_evaluation_metrics(y_true_lsa, y_pred_lsa, y_scores_lsa, "Random Forest␣
 ↪LSA")

# Plot metrics for Informative Topics
plot_evaluation_metrics(y_true_info, y_pred_info, y_scores_info, "Random Forest␣
 ↪Informative")
```

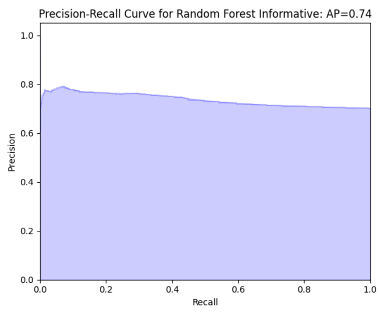












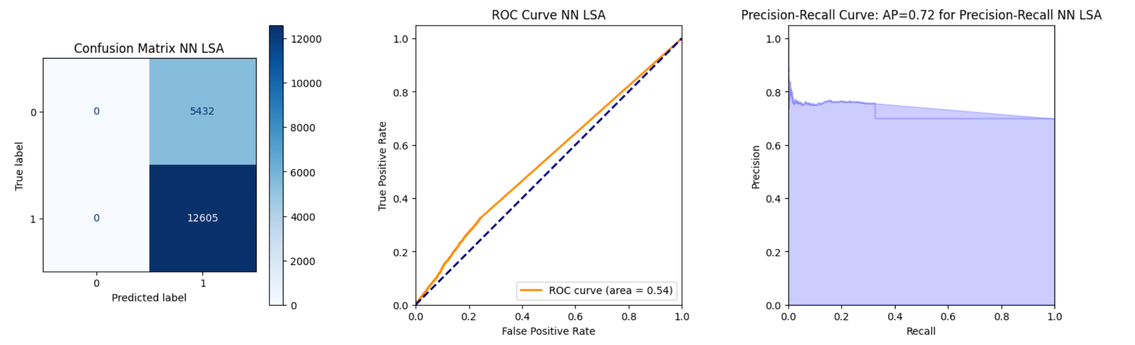
[26]:
```
# Define a function to perform cross-validation and return average accuracy
def perform_cv(pipeline, X, y, cv=5, scoring='accuracy'):
    # Applying cross-validation
    cv_scores = cross_val_score(pipeline, X, y, cv=cv, scoring=scoring)

    # Return the mean of the cross-validation accuracy scores
    return cv_scores.mean()

# Neural Network - LSA Topics
pipeline_lsa_nn = make_pipeline(SimpleImputer(strategy='mean'),␣
 ↪MLPClassifier(hidden_layer_sizes=(100,), max_iter=500, random_state=42))
cv_accuracy_lsa_nn = perform_cv(pipeline_lsa_nn, X_train_lsa, y_train_lsa)
print("Neural Network LSA - CV Mean Accuracy:", cv_accuracy_lsa_nn)
print("\n")

# Neural Network - Informative Topics
pipeline_info_nn = make_pipeline(SimpleImputer(strategy='mean'),␣
 ↪MLPClassifier(hidden_layer_sizes=(100,), max_iter=500, random_state=42))
cv_accuracy_info_nn = perform_cv(pipeline_info_nn, X_train_info, y_train_info)
print("Neural Network Informative - CV Mean Accuracy:", cv_accuracy_info_nn)
```

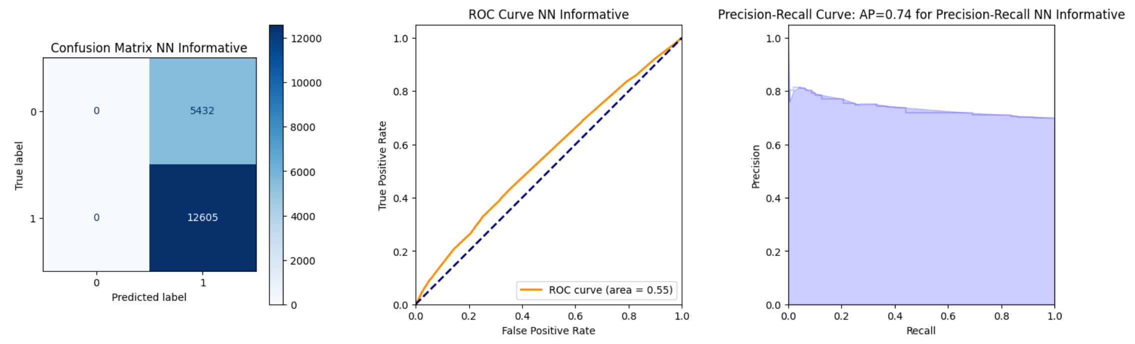
```
Neural Network LSA - CV Mean Accuracy: 0.701046503569201
Neural Network Informative - CV Mean Accuracy: 0.701046503569201
```

[27]:
```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay,␣
↪roc_curve, auc, precision_recall_curve, average_precision_score

def add_confusion_matrix(ax, y_true, y_pred, title):
    """Plot confusion matrix on a specific subplot axis."""
    cm = confusion_matrix(y_true, y_pred)
    disp = ConfusionMatrixDisplay(confusion_matrix=cm)
    disp.plot(ax=ax, cmap=plt.cm.Blues)
    ax.title.set_text(title)

def add_roc_curve(ax, y_true, y_scores, title):
    """Plot ROC curve on a specific subplot axis."""
    fpr, tpr, _ = roc_curve(y_true, y_scores)
    roc_auc = auc(fpr, tpr)
    ax.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area =␣
↪{roc_auc:.2f})')
    ax.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
    ax.set_xlim([0.0, 1.0])
    ax.set_ylim([0.0, 1.05])
    ax.set_xlabel('False Positive Rate')
    ax.set_ylabel('True Positive Rate')
    ax.legend(loc="lower right")
    ax.title.set_text(title)

def add_precision_recall_curve(ax, y_true, y_scores, title):
    """Plot precision-recall curve on a specific subplot axis."""
    precision, recall, _ = precision_recall_curve(y_true, y_scores)
    ap_score = average_precision_score(y_true, y_scores)
    ax.step(recall, precision, color='b', alpha=0.2, where='post')
    ax.fill_between(recall, precision, alpha=0.2, color='b')
    ax.set_xlabel('Recall')
    ax.set_ylabel('Precision')
    ax.set_ylim([0.0, 1.05])
    ax.set_xlim([0.0, 1.0])
    ax.title.set_text(f'Precision-Recall Curve: AP={ap_score:.2f} for {title}')
```

[28]:
```
import matplotlib.pyplot as plt
from sklearn.pipeline import make_pipeline
from sklearn.impute import SimpleImputer
from sklearn.neural_network import MLPClassifier

# Assuming models are trained; fit them if not already done
pipeline_lsa_nn = make_pipeline(SimpleImputer(strategy='mean'),␣
↪MLPClassifier(hidden_layer_sizes=(100,), max_iter=500, random_state=42))
pipeline_lsa_nn.fit(X_train_lsa, y_train_lsa)
```

```
pipeline_info_nn = make_pipeline(SimpleImputer(strategy='mean'),␣
 ↪MLPClassifier(hidden_layer_sizes=(100,), max_iter=500, random_state=42))
pipeline_info_nn.fit(X_train_info, y_train_info)

# Prepare the 2x3 grid of plots
fig, axes = plt.subplots(nrows=2, ncols=3, figsize=(18, 12))
plt.subplots_adjust(hspace=0.4, wspace=0.4)

# Neural Network LSA Topics
y_pred_lsa_nn = pipeline_lsa_nn.predict(X_test_lsa)
y_scores_lsa_nn = pipeline_lsa_nn.predict_proba(X_test_lsa)[:, 1]
add_confusion_matrix(axes[0, 0], y_test_lsa, y_pred_lsa_nn, 'Confusion Matrix␣
 ↪NN LSA')
add_roc_curve(axes[0, 1], y_test_lsa, y_scores_lsa_nn, 'ROC Curve NN LSA')
add_precision_recall_curve(axes[0, 2], y_test_lsa, y_scores_lsa_nn,␣
 ↪'Precision-Recall NN LSA')

# Neural Network Informative Topics
y_pred_info_nn = pipeline_info_nn.predict(X_test_info)
y_scores_info_nn = pipeline_info_nn.predict_proba(X_test_info)[:, 1]
add_confusion_matrix(axes[1, 0], y_test_info, y_pred_info_nn, 'Confusion Matrix␣
 ↪NN Informative')
add_roc_curve(axes[1, 1], y_test_info, y_scores_info_nn, 'ROC Curve NN␣
 ↪Informative')
add_precision_recall_curve(axes[1, 2], y_test_info, y_scores_info_nn,␣
 ↪'Precision-Recall NN Informative')

plt.show()
```





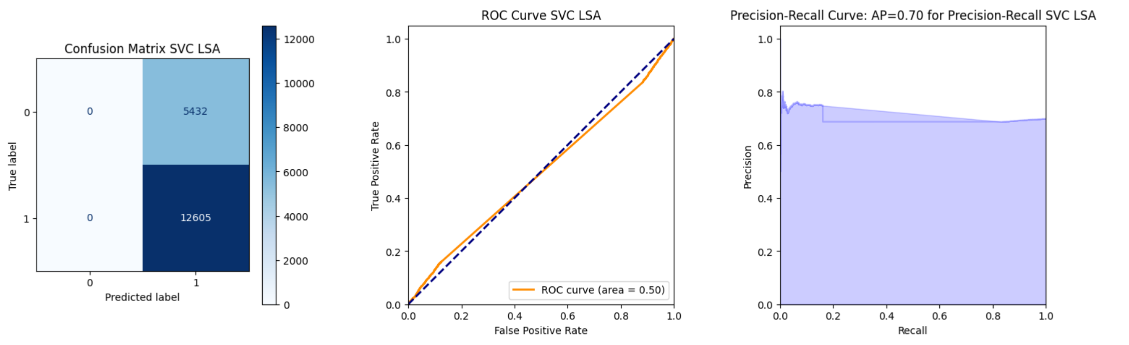
[29]:
```
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.svm import SVC

pipeline_lsa = Pipeline([
    ('imputer', SimpleImputer(strategy='mean')),
    ('svc', SVC(kernel='linear'))
])

pipeline_info = Pipeline([
    ('imputer', SimpleImputer(strategy='mean')),
    ('svc', SVC(kernel='linear'))
])
```

[30]:
```
# Fit the pipelines
pipeline_lsa.fit(X_train_lsa, y_train_lsa)
pipeline_info.fit(X_train_info, y_train_info)
```

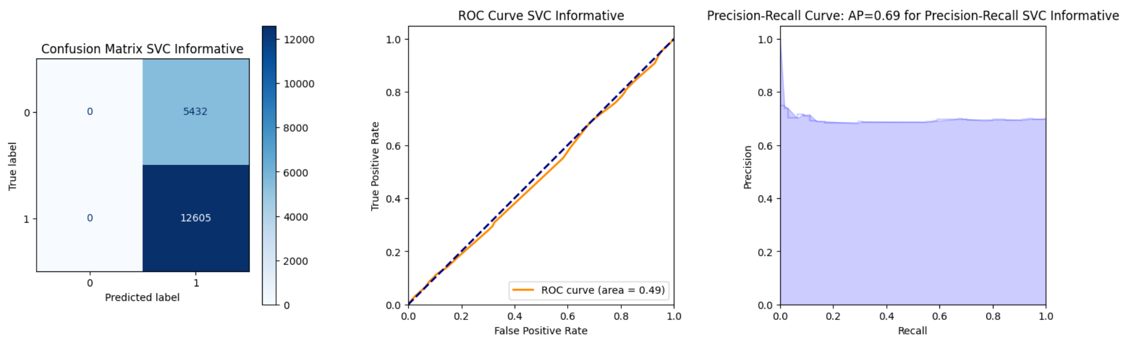
[30]:
```
Pipeline(steps=[('imputer', SimpleImputer()), ('svc', SVC(kernel='linear'))])
```

[34]:
```
from sklearn.pipeline import make_pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
```

```
pipeline_lsa_svc = make_pipeline(
    SimpleImputer(strategy='mean'),
    StandardScaler(),
    SVC(kernel='linear', probability=True, random_state=42)
)
pipeline_info_svc = make_pipeline(
    SimpleImputer(strategy='mean'),
    StandardScaler(),
    SVC(kernel='linear', probability=True, random_state=42)
)

pipeline_lsa_svc.fit(X_train_lsa, y_train_lsa)
pipeline_info_svc.fit(X_train_info, y_train_info)

fig, axes = plt.subplots(nrows=2, ncols=3, figsize=(18, 12))
plt.subplots_adjust(hspace=0.4, wspace=0.4)

# Predictions and evaluation for LSA SVC
y_pred_lsa_svc = pipeline_lsa_svc.predict(X_test_lsa)
y_scores_lsa_svc = pipeline_lsa_svc.predict_proba(X_test_lsa)[:, 1]
add_confusion_matrix(axes[0, 0], y_test_lsa, y_pred_lsa_svc, 'Confusion Matrix␣
 ↪SVC LSA')
add_roc_curve(axes[0, 1], y_test_lsa, y_scores_lsa_svc, 'ROC Curve SVC LSA')
add_precision_recall_curve(axes[0, 2], y_test_lsa, y_scores_lsa_svc,␣
 ↪'Precision-Recall SVC LSA')

# Predictions and evaluation for Informative SVC
y_pred_info_svc = pipeline_info_svc.predict(X_test_info)
y_scores_info_svc = pipeline_info_svc.predict_proba(X_test_info)[:, 1]
add_confusion_matrix(axes[1, 0], y_test_info, y_pred_info_svc, 'Confusion␣
 ↪Matrix SVC Informative')
add_roc_curve(axes[1, 1], y_test_info, y_scores_info_svc, 'ROC Curve SVC␣
 ↪Informative')
add_precision_recall_curve(axes[1, 2], y_test_info, y_scores_info_svc,␣
 ↪'Precision-Recall SVC Informative')

plt.show()
```



Confusion Matrix SVC LSA
0
5432
0
12605
True label
Predicted label
ROC Curve SVC LSA
True Positive Rate
False Positive Rate
ROC curve (area = 0.50)
Precision-Recall Curve: AP=0.70 for Precision-Recall SVC LSA
Precision
Recall



Confusion Matrix SVC Informative
0
5432
0
12605
True label
Predicted label
ROC Curve SVC Informative
True Positive Rate
False Positive Rate
ROC curve (area = 0.49)
Precision-Recall Curve: AP=0.69 for Precision-Recall SVC Informative
Precision
Recall

[ ]:

[ ]: