Consumer Review Analysis on Clothing Products

Group 01

Members:

- EG/2020/4333 - Shahmi A.J.A
- EG/2020/4041 - Laksayan T

---

01) Introduction

Our project focuses on analyzing consumer reviews of clothing products using machine learning techniques. The dataset, sourced from Kaggle, contains detailed customer insights, including review titles, full reviews, ratings, clothing categories, material types, construction quality, color, finishing, and durability.

The primary goal of this project is to develop a supervised classification model to predict customer ratings based on these features. By leveraging Support Vector Machines (SVM) and K-Nearest Neighbours (KNN) algorithms, we aim to extract meaningful patterns from the reviews and enhance the accuracy of rating predictions.

This project not only contributes to the field of sentiment analysis in fashion retail but also showcases the power of machine learning in understanding consumer preferences and improving product offerings.

---

02) Literature Survey

Online reviews have become an important source of information for customers and retailers. They provide insights into product quality, design, and customer satisfaction. Since the number of reviews is very large, machine learning techniques are widely used to analyze them.

Sentiment analysis, or opinion mining, helps in understanding whether reviews are positive, negative, or neutral. Studies have shown that algorithms like Support Vector Machines (SVM) and K-Nearest Neighbours (KNN) work well for text classification. SVM is effective in handling high-dimensional data, while KNN is simple and useful in finding similarities between reviews.

Most research focuses only on review text, but adding product-related features such as category, material, and durability can improve prediction accuracy. Previous studies also highlight that online reviews strongly influence consumer purchasing decisions, which makes predicting ratings very valuable.

This project addresses a gap in research by analyzing clothing product reviews using both text and product features. By applying SVM and KNN, the study aims to predict consumer ratings (1–5) and provide insights that can help fashion retailers improve their products and customer experience.

https://pdfs.semanticscholar.org/36aa/69afa98934c796b4a7bb1a5b5dffdaa29586.pdf

https://www.researchgate.net/publication/385511331_E-commerce-Clothing-Review-Analysis-by-Advanced-ML-Algorithms

---

03) Dataset Description

The dataset is collected on our own from various sources. This dataset comprises a comprehensive collection of reviews pertaining to clothing products and serves as a valuable resource for multilabel classification research. Each data entry is meticulously annotated with relevant labels, allowing researchers to explore various dimensions of the clothing products being reviewed. The dataset offers a rich diversity of perspectives and opinions, enabling the development and evaluation of robust classification models that can accurately predict multiple aspects of a clothing item. With its focus on multilabel classification, this data contributes significantly to advancing the understanding and application of machine learning algorithms in the fashion industry.

https://www.kaggle.com/datasets/jocelyndumlao/consumer-review-of-clothing-product

---

04) Exploratory Analysis

[ ] ↳ 13 cells hidden

05) Data Preprocessing

Handling NullValues

```
#finding the correlation between numerical features
df[[ 'Cons_rating', 'Materials','Construction', 'Color', 'Finishing', 'Durability']].corr()
```

	Cons_rating	Materials	Construction	Color	Finishing	Durability
Cons_rating	1.000000	0.012183	0.134932	0.119464	0.205068	-0.380981
Materials	0.012183	1.000000	-0.049311	0.100083	0.039825	0.001718
Construction	0.134932	-0.049311	1.000000	-0.098163	-0.049273	-0.307655
Color	0.119464	0.100083	-0.098163	1.000000	0.033431	-0.037341
Finishing	0.205068	0.039825	-0.049273	0.033431	1.000000	-0.076551
Durability	-0.380981	0.001718	-0.307655	-0.037341	-0.076551	1.000000

```
# here we can see that the correlation for between Cons_rating and other variable. those are
# therefore we can drop those columns
```

```
columns_to_drop = ['Materials', 'Construction', 'Color', 'Finishing', 'Durability']
df = df.drop(columns=columns_to_drop)
```

```
df.head()
```

	Title	Review	Cons_rating	Cloth_class
0	NaN	Absolutely wonderful - silky and sexy and comf...	4.0	Intimates
1	NaN	Love this dress! it's sooo pretty. i happene...	5.0	Dresses
2	Some major design flaws	I had such high hopes for this dress and reall...	3.0	Dresses
		I love, love, love this jumpsuit. it's fun,		

Next steps: Generate code with df View recommended plots New interactive sheet

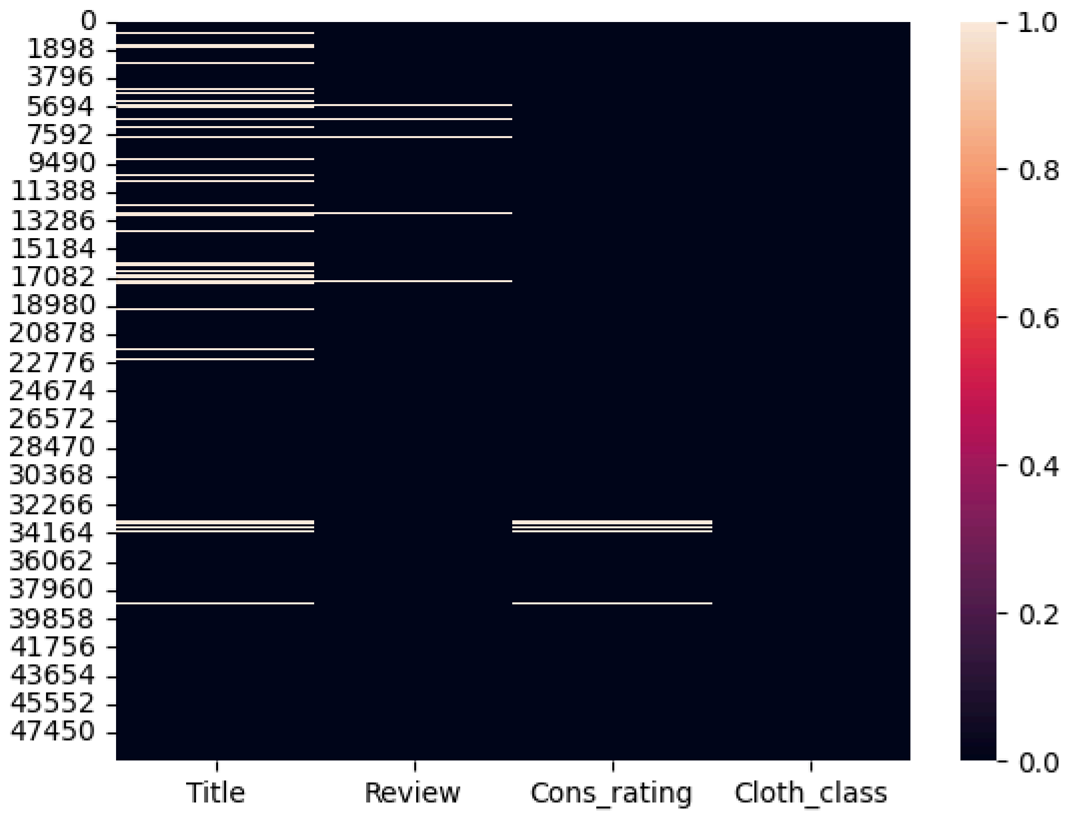
```
# sum of the missing values
df.isnull().sum()
```

	0
Title	3968
Review	831
Cons_rating	214
Cloth_class	16

dtype: int64

```
#plotting the heatmap. white color defines the nul values
sns.heatmap(data=df.isnull())
```

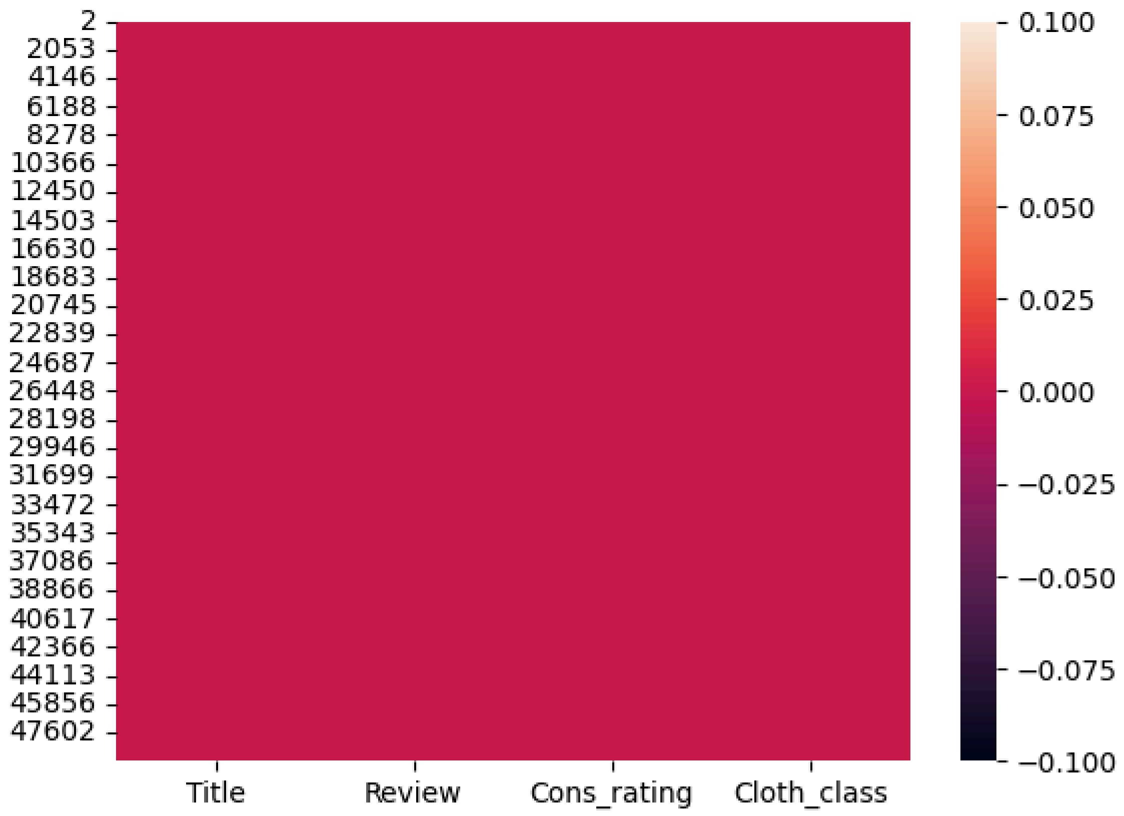
```
<Axes: >
```



```
# dropping the missing values because missing values are small compared to the dataset
df = df.dropna()
```

```
# plotting the heatmap again. here we can see that there is no null values
sns.heatmap(data=df.isnull())
```

```
<Axes: >
```



```
df.isnull().sum()
```

	0
Title	0
Review	0
Cons_rating	0
Cloth_class	0

dtype: int64

```
df.shape
```

```
(45308, 4)
```

```
df.describe()
```

	Cons_rating
count	45308.000000
mean	4.086541
std	1.296556
min	1.000000
25%	3.000000
50%	5.000000
75%	5.000000
max	5.000000

```
df['Cons_rating'].value_counts()
```

	count
Cons_rating	
5.0	26097
4.0	7483
3.0	4947
1.0	3667
2.0	3114

dtype: int64

```
df['Cloth_class'].value_counts()
```

Cloth_class	count
Dresses	7639
Blouses	5042
Knits	3981
Jeans	3772
Sweaters	3638
Pants	3436
Jackets	3114
Shorts	3021
Sleep	2722
Shirts	2498
Blazer	1768
Suits	1309
Fine gauge	927
Skirts	796
Lounge	574
Swim	293
Outerwear	281
Legwear	131
Intimates	120
Layering	115
Trend	107
Dress	22
Chemises	1
Casual bottoms	1

dtype: int64

```
# as we are interested in review and ratings we can drop cloth_class and Title
df=df.drop(columns=['Title','Cloth_class'])
```

```python
df.head()
```

	Review	Cons_rating
2	I had such high hopes for this dress and reall...	3.0
3	I love, love, love this jumpsuit. it's fun, fl...	5.0
4	This shirt is very flattering to all due to th...	5.0
5	I love tracy reese dresses, but this one is no...	2.0
6	I aded this in my basket at hte last mintue to...	5.0

Next steps: Generate code with df View recommended plots New interactive sheet

Data Cleaning

```python
# Import nltk for text preprocessing
import nltk

# Download necessary NLTK resources:
nltk.download('wordnet')        # WordNet corpus for lemmatization
nltk.download('stopwords')      # Common stopwords (e.g., 'the', 'is', 'and')
nltk.download('punkt_tab')          # Pre-trained tokenizer models

# Import stopwords, tokenizer, and lemmatizer
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer

# Install gensim (topic modeling, word embeddings, etc.)
!pip install gensim

# Import gensim after installation
import gensim

# Import additional utilities for text cleaning
import string   # to remove punctuation
import re       # to handle regular expressions (remove unwanted characters)
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet 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 punkt_tab to /root/nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
Requirement already satisfied: gensim in /usr/local/lib/python3.12/dist-packages (4.3.3)
Requirement already satisfied: numpy<2.0,>=1.18.5 in /usr/local/lib/python3.12/dist-pack
Requirement already satisfied: scipy<1.14.0,>=1.7.0 in /usr/local/lib/python3.12/dist-pa
Requirement already satisfied: smart-open>=1.8.1 in /usr/local/lib/python3.12/dist-packa
```

```
    Requirement already satisfied: wrapt in /usr/local/lib/python3.12/dist-packages (from sm
```

```
# 1. convert all the words into lowercase and removing punctuations
def clean_data1(text):
  text = [word.lower() for word in text]
  no_punct=[letter for letter in text if letter not in string.punctuation]
  words_wo_punct=''.join(no_punct)
  return words_wo_punct
df['cleaned_review_1']=df['Review'].apply(lambda x: clean_data1(x))
df.head()
```

	Review	Cons_rating	cleaned_review_1
2	I had such high hopes for this dress and reall...	3.0	i had such high hopes for this dress and reall...
3	I love, love, love this jumpsuit. it's fun, fl...	5.0	i love love love this jumpsuit its fun flirty ...
4	This shirt is very flattering to all due to th...	5.0	this shirt is very flattering to all due to th...
5	I love tracy reese dresses, but this		i love tracy reese dresses but this one is

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
# 2. removing unnecessary letters and brackets
def clean_data2(text):
  text = [re.sub(r"[0-9]", "",words) for words in text]          #removing numbers
  text = [re.sub(r"(\(.*\))|(\[.*\])", "",words) for words in text] #removing brackets
  text = [re.sub(r"[^\w\s]", "", words) for words in text]        #removing symbols
  text = ''.join(text)
  return text

df['cleaned_review_2']=df['cleaned_review_1'].apply(lambda x: clean_data2(x))
df.head()
```

	Review	Cons_rating	cleaned_review_1	cleaned_review_2
2	I had such high hopes for this dress and reall...	3.0	i had such high hopes for this dress and reall...	i had such high hopes for this dress and reall...
3	I love, love, love this jumpsuit. it's fun, fl...	5.0	i love love love this jumpsuit its fun flirty ...	i love love love this jumpsuit its fun flirty ...
4	This shirt is very flattering to all due	5.0	this shirt is very flattering	this shirt is very flattering

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
# 3. tokenize the strings (splitting the string into words) and removing stop words (is, are

def clean_data3(text):
  STOPWORDS = set(stopwords.words('english'))
  STOPWORDS.remove('not')                        # because "not" word is used for showing
  tokenized = word_tokenize(text)                # also we can use tokenized=re.split("\W
  cleaned_wo_sw = [word for word in tokenized if word not in STOPWORDS]
  return cleaned_wo_sw

df['cleaned_review_3']=df['cleaned_review_2'].apply(lambda x: clean_data3(x))
df.head()
```

	Review	Cons_rating	cleaned_review_1	cleaned_review_2	cleaned_review_3
2	I had such high hopes for this dress and reall...	3.0	i had such high hopes for this dress and reall...	i had such high hopes for this dress and reall...	[high, hopes, dress, really, wanted, work, ini...
3	I love, love, love this jumpsuit	5.0	i love love love this jumpsuit its fun flirty	i love love love this jumpsuit its fun flirty	[love, love, love, jumpsuit, fun, flirty,

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
# 4.Lemmatization (converts the words into base form eg: believes -> belief)
def clean_data4(text):
  lemmatizer = WordNetLemmatizer()
  lemmatized_data = [lemmatizer.lemmatize(word) for word in text]
  return lemmatized_data

df['cleaned_review_4']=df['cleaned_review_3'].apply(lambda x: clean_data4(x))
df.head()
```

	Review	Cons_rating	cleaned_review_1	cleaned_review_2	cleaned_review_3	cleaned_
2	I had such high hopes for this dress and reall...	3.0	i had such high hopes for this dress and reall...	i had such high hopes for this dress and reall...	[high, hopes, dress, really, wanted, work, ini...	[high, hc reall v
3	I love, love, love this jumpsuit. it's fun.	5.0	i love love love this jumpsuit its fun flirty ...	i love love love this jumpsuit its fun flirty ...	[love, love, love, jumpsuit, fun, flirty, fabu...	[love, jumpsuit,

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
# cleaned_review4 column is replaced the review
df['Review']=df['cleaned_review_4']
```

```
df=df.drop(columns=['cleaned_review_1','cleaned_review_2','cleaned_review_3','cleaned_reviev
df.head()
```

	Review	Cons_rating
2	[high, hope, dress, really, wanted, work, init...	3.0
3	[love, love, love, jumpsuit, fun, flirty, fabu...	5.0
4	[shirt, flattering, due, adjustable, front, ti...	5.0
5	[love, tracy, reese, dress, one, not, petite, ...	2.0
6	[aded, basket, hte, last, mintue, see, would, ...	5.0

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
def categorize_rating(rating):
    if rating > 3:
        return 'Positive'
    elif rating < 2 :
        return 'Negative'
    else:
        return 'Neutral'

# Apply the categorize_rating function to categorize the rating as negative(1,2),positive(3,
df['Cons_rating'] = df['Cons_rating'].apply(lambda x: categorize_rating(x))
```

```
# we have to joined the words as sentences because vectorization cannot be done with the arr
df['Review'] = [' '.join(text) for text in df['Review'] ]
```

```
df.head()
```

	Review	Cons_rating
2	high hope dress really wanted work initially o...	Neutral
3	love love love jumpsuit fun flirty fabulous ev...	Positive
4	shirt flattering due adjustable front tie perf...	Positive
5	love tracy reese dress one not petite foot tal...	Neutral
6	aded basket hte last mintue see would look lik...	Positive

Next steps: Generate code with df   View recommended plots   New interactive sheet

```
#making copies of df
df2 = df.copy()
df3 = df.copy()
```

06) Model Implementation

Using td idf vectorization.

```
# tf idf vectorizing . this converts the words into binary values
from sklearn.feature_extraction.text import TfidfVectorizer
vect = TfidfVectorizer(min_df = 5,
                             max_df = 0.8,
                             sublinear_tf = True,
                             use_idf = True)


X = df['Review']
Y = df['Cons_rating']


from sklearn.model_selection import train_test_split

# spliytting the dataset. 80 % for training and others for testing
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=42)


print("Size of x_train:", (x_train.shape))
print("Size of y_train:", (y_train.shape))
print("Size of x_test:", (x_test.shape))
print("Size of y_test:", (y_test.shape))
```

```
Size of x_train: (36246,)
Size of y_train: (36246,)
Size of x_test: (9062,)
Size of y_test: (9062,)
```

```
# Fit the vectorizer on the training data and transform it into a document-term matrix
x_train = vect.fit_transform(x_train)

# Transform the test data using the same vocabulary (no fitting again!)
x_test = vect.transform(x_test)
```

Train SVM and Generate Accuracy

```
# using svm to train the model
from sklearn import svm
import seaborn as sns
from sklearn.metrics import classification_report, accuracy_score
# Perform classification with SVM, kernel=linear

classifier_linear = svm.SVC(kernel='linear')

classifier_linear.fit(x_train, y_train)

prediction_linear = classifier_linear.predict(x_test)

report = classification_report(y_test, prediction_linear, output_dict=True)

# Extract Accuracy F1 score and precision from the classification report
accuracy = report['accuracy']
f1_score = report['weighted avg']['f1-score']
precision = report['weighted avg']['precision']

print("Accuracy:", accuracy)
print("F1 Score:", f1_score)
print("Precision:", precision)
```

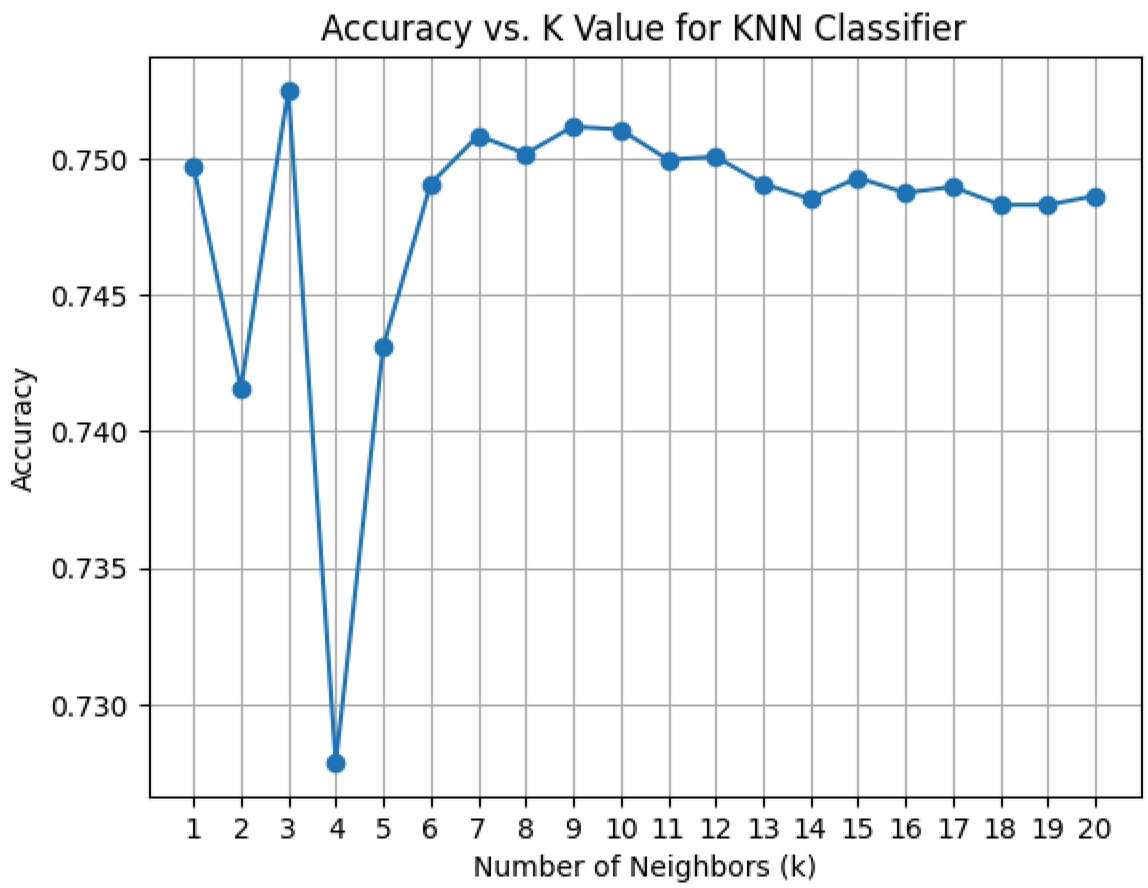
```
Accuracy: 0.8266387111013022
F1 Score: 0.8161258240413571
Precision: 0.8112359550192111
```

```
# Print the number (or score) of positive sentiments from the report
print('Positive: ', report['Positive'])

# Print the number (or score) of negative sentiments from the report
print('Negative: ', report['Negative'])

# Print the number (or score) of neutral sentiments from the report
print('Neutral: ', report['Neutral'])
```

```
Positive:  {'precision': 0.8892580287929125, 'recall': 0.9504364550969078, 'f1-score': 0
Negative:  {'precision': 0.608, 'recall': 0.41530054644808745, 'f1-score': 0.4935064935
Neutral:  {'precision': 0.570254110612855, 'recall': 0.4856779121578612, 'f1-score': 0.5
```

Train KNN and Generate Accuracy

```
# using knn
from sklearn.neighbors import KNeighborsClassifier

# for finding the optimal k value this graph is plotted
k_values = np.arange(1, 21)

# Initialize an empty list to store accuracy scores for each k
accuracy_scores = []

# Loop through different k values
for k in k_values:
    # Train a KNN classifier with the current k value
    knn_classifier = KNeighborsClassifier(n_neighbors=k)
    knn_classifier.fit(x_train, y_train)

    # Make predictions on the test set
    predictions = knn_classifier.predict(x_test)

    # Calculate accuracy and store it in the list
    accuracy = accuracy_score(y_test, predictions)
    accuracy_scores.append(accuracy)

# Plot the accuracy scores for different k values
plt.plot(k_values, accuracy_scores, marker='o')
plt.title('Accuracy vs. K Value for KNN Classifier')
plt.xlabel('Number of Neighbors (k)')
plt.ylabel('Accuracy')
plt.xticks(k_values)
plt.grid(True)
plt.show()
```



```
# using knn
from sklearn.neighbors import KNeighborsClassifier

knn_classifier = KNeighborsClassifier(n_neighbors=5)
knn_classifier.fit(x_train, y_train)

predictions = knn_classifier.predict(x_test)
```

```
Accuracy: 0.7431030677554624
```

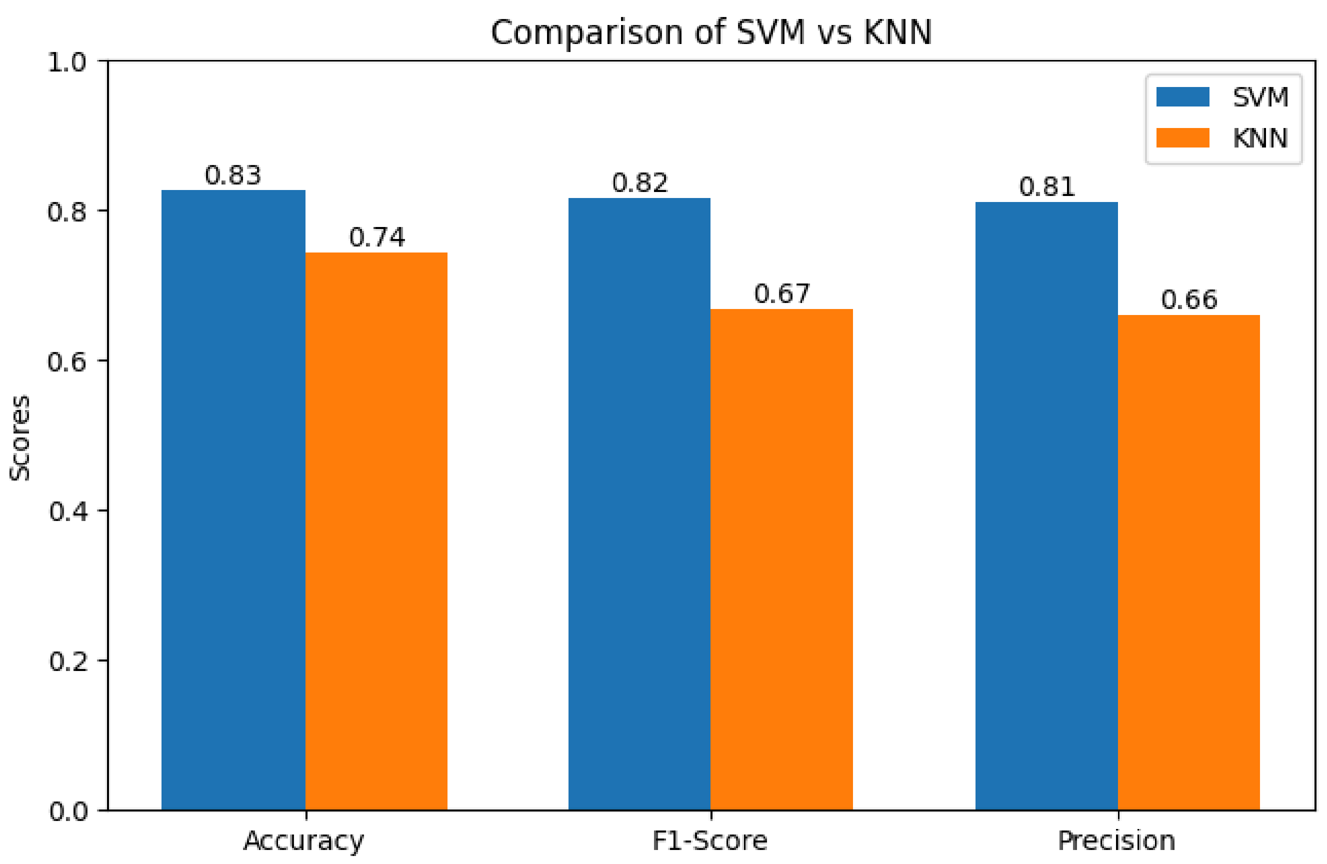
```
report2 = classification_report(y_test, predictions, output_dict=True)
# Extract Accuracy F1 score and precision from the classification report
accuracy2 = report2['accuracy']
f1_score2 = report2['weighted avg']['f1-score']
precision2 = report2['weighted avg']['precision']

print("Accuracy:", accuracy)
print("F1 Score:", f1_score)
print("Precision:", precision)
```

```
Accuracy: 0.7431030677554624
F1 Score: 0.6674794117105669
Precision: 0.6607074073384134
```

07) Model Evaluation and Discussion

Accuracy Comparision

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import accuracy_score, f1_score, precision_score

# Calculate metrics
svm_acc = accuracy_score(y_test, prediction_linear)
svm_f1 = f1_score(y_test, prediction_linear, average='weighted')
svm_prec = precision_score(y_test, prediction_linear, average='weighted')

knn_acc = accuracy_score(y_test, predictions)
knn_f1 = f1_score(y_test, predictions, average='weighted')
knn_prec = precision_score(y_test, predictions, average='weighted')

# Prepare data for plotting
metrics = ['Accuracy', 'F1-Score', 'Precision']
svm_scores = [svm_acc, svm_f1, svm_prec]
knn_scores = [knn_acc, knn_f1, knn_prec]

x = np.arange(len(metrics))
width = 0.35

fig, ax = plt.subplots(figsize=(8,5))
rects1 = ax.bar(x - width/2, svm_scores, width, label='SVM')
rects2 = ax.bar(x + width/2, knn_scores, width, label='KNN')

ax.set_ylabel('Scores')
ax.set_title('Comparison of SVM vs KNN')
ax.set_xticks(x)
ax.set_xticklabels(metrics)
ax.legend()
ax.bar_label(rects1, fmt='%.2f')
ax.bar_label(rects2, fmt='%.2f')

plt.ylim(0, 1)
plt.show()
```



Predict output with newdata SVM

```
new_text = ["i like this dress"]

new_text = clean_data1(new_text);
new_text = clean_data2(new_text);
new_text = clean_data3(new_text);
new_text = clean_data4(new_text);
# Transform the new text using the same vectorizer
new_text_vectorized = vect.transform(new_text);

# Use the trained KNN model to predict the category for the new text
predicted_category = classifier_linear.predict(new_text_vectorized)

# Display the predicted category
print("Predicted Category:", predicted_category[0])
```

```
Predicted Category: Positive
```

Predict output with newdata KNN

```
new_text = ["i like this dress"]

new_text = clean_data1(new_text);
new_text = clean_data2(new_text);
new_text = clean_data3(new_text);
new_text = clean_data4(new_text);
# Transform the new text using the same vectorizer
new_text_vectorized = vect.transform(new_text);

# Use the trained KNN model to predict the category for the new text
predicted_category = knn_classifier.predict(new_text_vectorized)

# Display the predicted category
print("Predicted Category:", predicted_category[0])
```

```
Predicted Category: Positive
```

Discussion