Infosys Springboard Internship

Domain: Artificial Intelligence(AI)

Project Title:

AI STYLIST

Team Members:

- Akshay
- Dheeraj
- Lebin Bright
- Mohammed Ghouse
- Shameera Begum

Mentor:
Anil Shaw Sir

1.DatasetOverview

- **FileName:**fashion_products_data.ldjson
- **Format:**JSONLines(.ldjson)
- **Purpose:**Likelycontainsproduct-relateddata,typicallyusedfor fashionindustryanalysis.

2.DataframeSummary

- **Shape(Rows,Columns):**Tobedetermineduspondata inspection.
- **Columns:**Thedataframecontainsthefollowingcolumns(columnnamestobe extracted):
  - Example:product_id,name,category,price,stock_status.
- **ColumnData Types:**
  Example*:*
  - product_id→Integer
  - name→String
  - price→Float
- **NullValues:**
  Numberofmissingvalues ineachcolumn.

3.DescriptiveStatistics

- **NumericalColumns:**
  - Count,Mean,StandardDeviation,Min,Maxforcolumnslikeprice.
- **CategoricalColumns:**
  - Countofuniquecategories,mostfrequentvalues.

5.Insights

1. **DataCompleteness:**
   - Columnswithsignificantmissingdatashouldbeaddressed.
2. **PotentialAnalysis:**
   - Pricetrendsbycategory.
   - Stockavailabilityinsights.

DatasetInfo:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30000 entries, 0 to 29999
Data columns (total 33 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   uniq_id                         30000 non-null  object
 1   crawl_timestamp                 30000 non-null  object
 2   asin                            30000 non-null  object
 3   product_url                     30000 non-null  object
 4   product_name                    30000 non-null  object
 5   image_urls__small               29998 non-null  object
 6   medium                          29998 non-null  object
 7   large                           28841 non-null  object
 8   browsenode                      29480 non-null  float64
 9   brand                           21857 non-null  object
 10  sales_price                     27110 non-null  float64
 11  weight                          30000 non-null  object
 12  rating                          30000 non-null  float64
 13  sales_rank_in_parent_category   25497 non-null  object
 14  sales_rank_in_child_category    24851 non-null  object
 15  delivery_type                   30000 non-null  object
 16  meta_keywords                   30000 non-null  object
 17  amazon_prime__y_or_n            30000 non-null  object
 18  parent___child_category__all    25497 non-null  object
 19  best_seller_tag__y_or_n         30000 non-null  object
...
 31  formats___editions              2 non-null      object
 32  name_of_author_for_books        1 non-null      object
```

Dtypes:float64(7),object(26)

Memoryusage:7.6+MB

Introduction:

Inthe realmofdata analysis,particularlyconcerningfashionproductdatasets,severalcritical techniquesandmethodologiesareemployedtoensurethatthedataisbothinformativeand actionable.Thisreportdelves intovarioustopicscoveredintheJupyter Notebook, including **data visualization**, **outliers**, **null values**, **stop words**, **short forms**, **punctuation**, **non-English words**, **stemming**, and **lemmatization**.Eachsectiondefines keyconcepts,discussesrelevantlibrariesandfunctions,andevaluatestheirusecases, advantages,anddisadvantages.

	uniq_id	crawl_timestamp	asin	product_url	product_name	image_urls__small	medium
0	26d41bdc1495de290bc8e6062d927729	2020-02-07 05:11:36 +0000	B07STS2W9T	https://www.amazon.in/Facon-Kalamkari-Handbloc...	LA' Facon Cotton Kalamkari Handblock Saree Blo...	https://images-na.ssl-images-amazon.com/images...	https://images-na.ssl-images-amazon.com/images...
1	410c62298852e68f34c35560f2311e5a	2020-02-07 08:45:56 +0000	B07N6TD2WL	https://www.amazon.in/Sf-Jeans-Pantaloons-T-Sh...	Sf Jeans By Pantaloons Men's Plain Slim fit T-...	https://images-na.ssl-images-amazon.com/images...	https://images-na.ssl-images-amazon.com/images...
2	52e31bb31680b0ec73de0d781a23cc0a	2020-02-06 11:09:38 +0000	B07WJ6WPN1	https://www.amazon.in/LOVISTA-Traditional-Prin...	LOVISTA Cotton Gota Patti Tassel Traditional P...	https://images-na.ssl-images-amazon.com/images...	https://images-na.ssl-images-amazon.com/images...
3	25798d6dc43239c118452d1bee0fb088	2020-02-07 08:32:45 +0000	B07PYSF4WZ	https://www.amazon.in/People-Printed-Regular-T...	People Men's Printed Regular fit T-Shirt	https://images-na.ssl-images-amazon.com/images...	https://images-na.ssl-images-amazon.com/images...
4	ad8a5a196d515ef09dfdaf082bdc37c4	2020-02-06 14:27:48 +0000	B082KXNM7X	https://www.amazon.in/Monte-Carlo-Cotton-Colla...	Monte Carlo Grey Solid Cotton Blend Polo Colla...	https://images-na.ssl-images-amazon.com/images...	https://images-na.ssl-images-amazon.com/images...

1.Data Visualization:

Definition:

Datavisualizationisthegraphicalrepresentationofinformationanddata.Byusing visual elements likecharts,graphs,andmaps,datavisualizationtoolsprovideanaccessiblewaytosee andunderstandtrends,outliers,andpatterns indata.

Librariesand Functions

1. **Matplotlib**

- **Function**:plt.plot(),plt.bar(),plt.pie()
- **UseCase**: Creatingstatic graphsforexploratorydataanalysis.
- **Advantages**:Highlycustomizable;supportsawide range ofplots.
- **Disadvantages**:Steeperlearningcurvefor complexvisualizations.

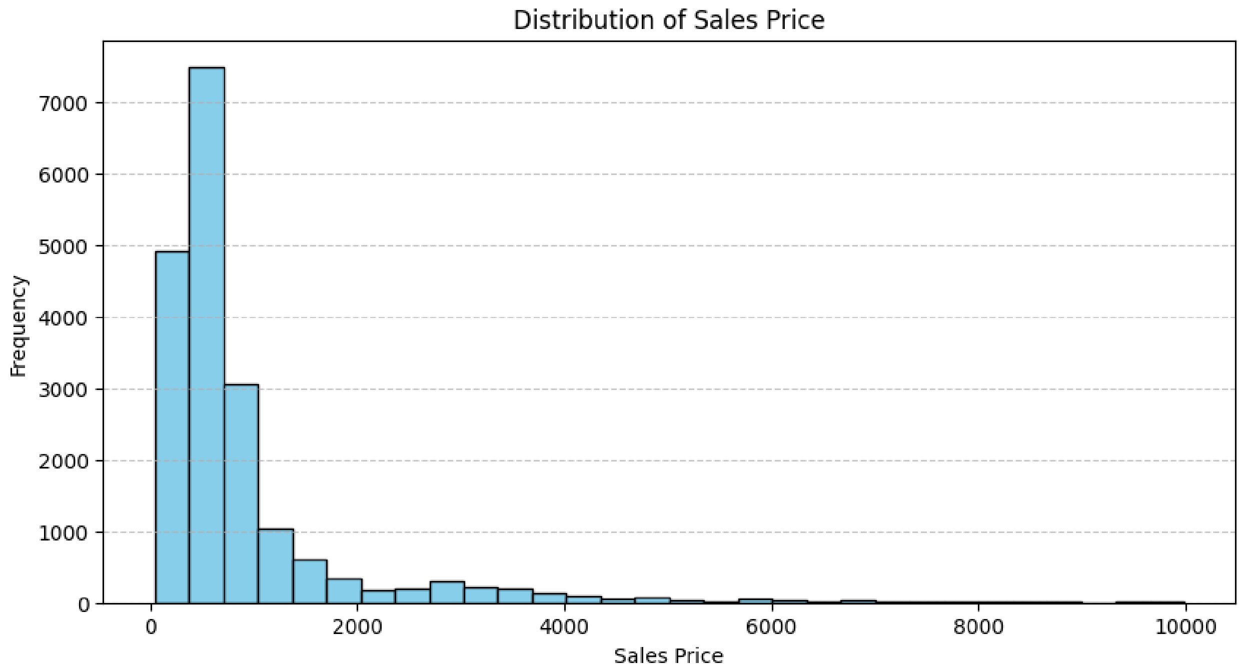
2. **Seaborn**

- **Function**:sns.scatterplot(),sns.boxplot()
- **UseCase**: Statisticaldatavisualizationwithattractivedefaults.
- **Advantages**:Simplifiescomplexvisualizations; integrateswellwithpandas.

- **Disadvantages**:Lessflexible thanMatplotlib forcertaintypesofplots.

```
importpandasaspd
importmatplotlib.pyplotasplt
import json

#Loadthedatafromthe.ldjson file
file_path='C:/Users/admin/fashion_products_data.ldjson'#Updatedpath data =
[]

#Openandreaddatalinebyline
withopen(file_path,'r',encoding='utf-8')asfile: for
    line in file:
        data.append(json.loads(line))

#ConvertdataintoaDataFrame df =
pd.DataFrame(data)

#DataCleaningandPreparation
# Convert sales_price and rating columns to numeric, handling errors
df['sales_price']=pd.to_numeric(df['sales_price'],errors='coerce')
df['rating'] = pd.to_numeric(df['rating'], errors='coerce')

#DroprowswithNaNvaluesinessentialcolumns
df=df.dropna(subset=['product_url','sales_price','rating','brand'])

#BasicAnalysis
print("Summaryofthe'sales_price'column:") print(df['sales_price'].describe())

print("\nTop10brandsbynumberofproducts:")
print(df['brand'].value_counts().head(10))

#Visualization:DistributionofSalesPrice
plt.figure(figsize=(10, 5))
plt.hist(df['sales_price'].dropna(),bins=30,color='skyblue',edgecolor='black')
plt.title("DistributionofSalesPrice")
```

```
plt.xlabel("SalesPrice")
plt.ylabel("Frequency")
plt.grid(axis='y',linestyle='--',alpha=0.7)
plt.show()

# Visualization: Average Rating by Top 10 Brands
top_brands=df['brand'].value_counts().nlargest(10).index
df_top_brands = df[df['brand'].isin(top_brands)]
plt.figure(figsize=(12, 6))
box_data=[df_top_brands[df_top_brands['brand']==brand]['rating'].dropna()for
brand in top_brands]
plt.boxplot(box_data,labels=top_brands,patch_artist=True) plt.xticks(rotation=45)
plt.title("RatingDistributionbyTop10Brands")
plt.xlabel("Brand")
plt.ylabel("Rating")
plt.show()

#Visualization:RelationshipbetweenSalesPriceandRating plt.figure(figsize=(8, 5))
forbrandintop_brands:
    brand_data = df[df['brand'] == brand]
    plt.scatter(brand_data['sales_price'],brand_data['rating'],label=brand,
alpha=0.6)

plt.title("SalesPricevsRating")
plt.xlabel("Sales Price")
plt.ylabel("Rating")
plt.legend(title="Brand",bbox_to_anchor=(1.05,1),loc='upperleft')

plt.grid(linestyle='--',alpha=0.7)
plt.show()

#CheckingforOutliersinSalesPrice
outliers=df[df['sales_price']>df['sales_price'].quantile(0.99)]
print("\nPotential Sales Price Outliers:")
print(outliers[['product_url', 'sales_price', 'brand', 'rating']])
```

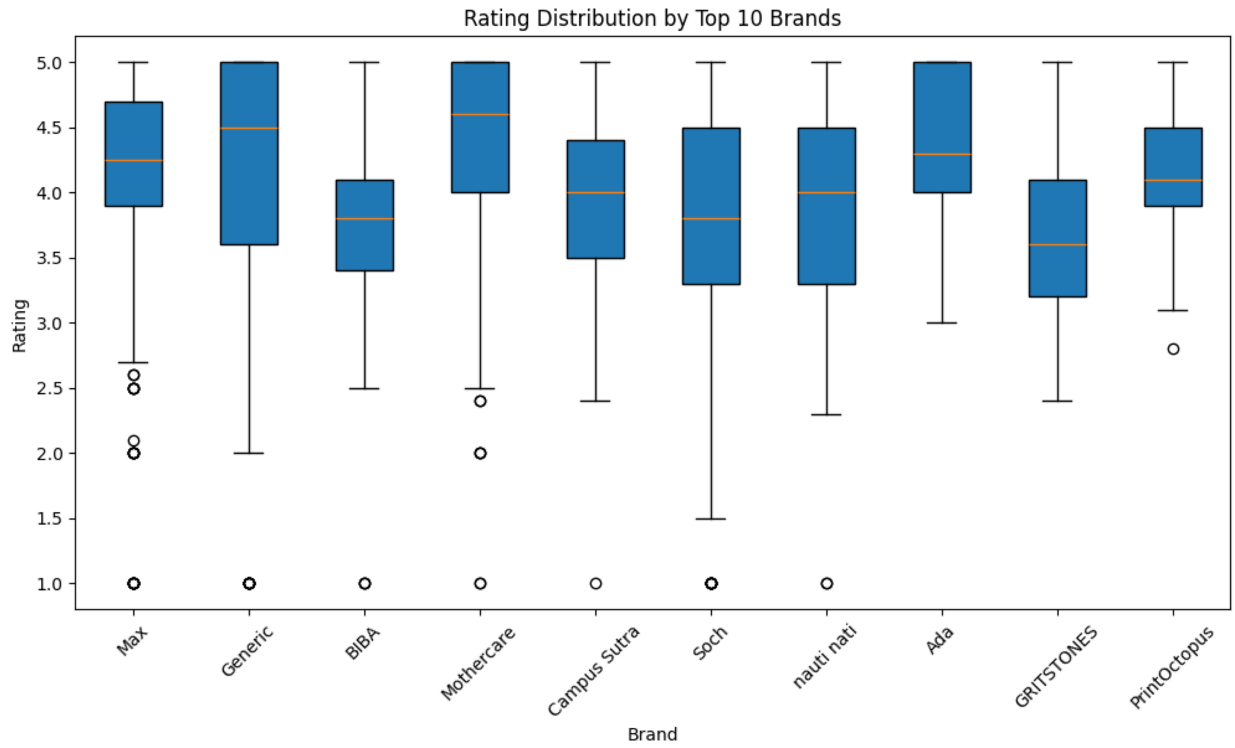


Histogram:

Thehistogramshowcasesthedistributionofsalesprices,revealing insightsintothefrequency of different price points.

KeyObservation:

- From this above Histogram, we understood that it is a **Right-Skewed** distribution because the majorityofsalesoccursat lowersprices,withthe frequencytapering off as the price increases.
- Thehighestfrequencyofsalesfallswithinthe$0 to $2000 range,**Peaking**around $1000.
- Asignificantnumberofsalesarealsopresentathigherpricepoints,extendingbeyond $8000.
- Thisdistributionsuggeststhattheproductorservicebeinganalyzed hasawiderangeof prices, appealing to both budget-conscious and higher-end consumers.



BoxPlot:

ThisBoxPlotRepresentsthe inter quartilerange(IQR),whichcontainsthe middle50%ofthe data. The bottomofthe box is the first quartile (25th percentile), and the top is the third quartile (75th percentile).

- **Thelineinsidethebox:**

  ThisIndicatesthemedian(50thpercentile),whichisthemiddlevalueofthe data.

- **Thedots:**

  Itrepresentoutliers,whicharedatapointsthatfalloutsidethewhiskers.

Outliers:

Definition:

Outliersaredatapointsthatdiffer significantly fromotherobservations.Theycanskewresults andleadto misleadinginterpretations ifnotaddressedproperly.

LibrariesandFunctions

- **Pandas**
- **Function**:df.describe(),df[df['column']> threshold]
- **UseCase**:Identifyingoutliersbasedonstatistical measureslikeIQRorZ-score.
- **Advantages**:Easyto implement;integratesseamlesslywithDataFrame operations.
- **Disadvantages**:Mayrequiremanualthresholdsetting.

ExampleCode:

```
Q1=df['Sales'].quantile(0.25)
Q3=df['Sales'].quantile(0.75)
IQR =Q3- Q1
outliers=df[(df['Sales']<(Q1-1.5*IQR))|(df['Sales']> (Q3+1.5*IQR))]
```

1. WeareThreeQuartiles:

- **Q1 (FirstQuartile):** Representsthe25thpercentileofthe Sales data.This isthevalue below which 25% of the data lies
- **Q2 (SecondQuartile):** Representsthe50thpercentileofthe Sales data.Thisisthe value below which 50% of the data lies.
- **Q3 (ThirdQuartile):** Representsthe75thpercentileofthe Sales data.Thisisthevalue below which 75% of the data lies.

2. ComputetheInterquartileRange(IQR):

- **IQR**:Measuresthespreadofthemiddle50% ofthedata.Itisthedifferencebetween Q3 and Q1.
- Formula:IQR=Q3−Q1\text{IQR}=Q3-Q1IQR=Q3−Q1

3. DefineOutlierBoundaries:

- **LowerBound**:AnydatapointlessthanQ1−1.5×IQRQ1-1.5\times\text{IQR}Q1−1.5×IQRis considered an outlier.

- **UpperBound**:Anydatapointgreater thanQ3+1.5×IQRQ3+1.5\times\text{IQR}Q3+1.5×IQR is considered an outlier.

Thefactor1.5isastandardmultiplierusedinstatisticstodefine**"mild"**outliers.

4.IdentifyOutliers:

- **Condition**:Thecodeusesbooleanindexingtofilter rowsintheSalescolumn:
  - RowswhereSalesislessthanQ1−1.5×IQRQ1-1.5 \times \text{IQR}Q1−1.5×IQR(extremelylowvalues).
  - RowswhereSalesisgreaterthanQ3+1.5×IQRQ3+ 1.5\times \text{IQR}Q3+1.5×IQR(extremelyhighvalues).
- **Output**:TheresultingoutliersDataFramecontains allrowswithoutliervalues inthe Salescolumn.

ScatterPlot:

Fromtheabovediagram, The**ScatterPlot** shows therelationship betweenSales PriceandRatingfor different brands.

- **Keypoints:**

  X-axis:

  ItRepresentstheSalesPrice.

  Y-axis:

  ItRepresentstheRating.

  Dots:

  Inthis, Eachdotrepresentsasingleproduct.

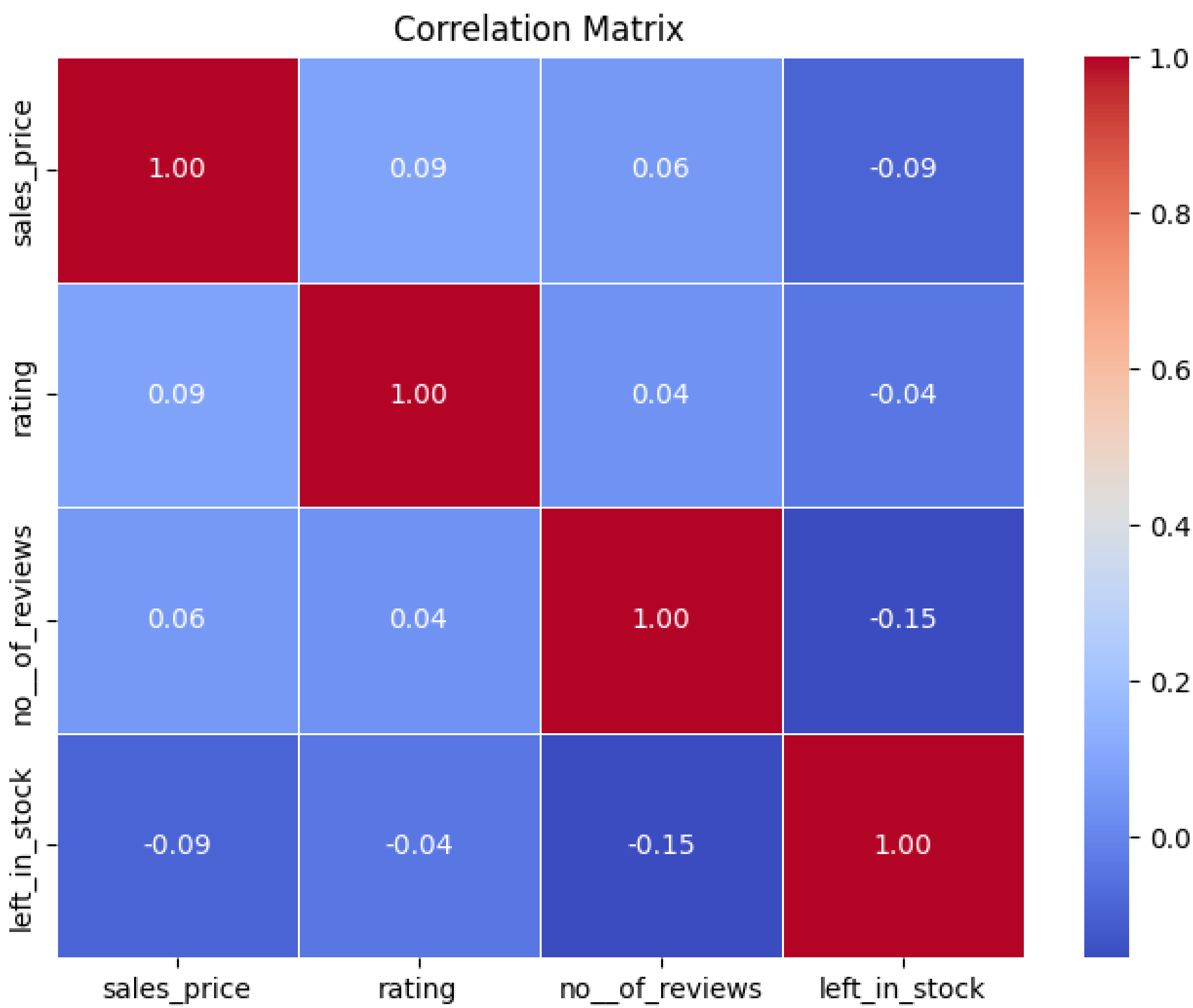
  Colors:

  ItShowsDifferentcolorsrepresentdifferentbrands.

  Interpretation:

  Theplotindicatesthatthereisn'ta strongcorrelationbetweenSalesPriceandRating.

- Someproductswithhighratingshavelower salesprices, whilesomewithlower ratingshave higher sales prices.
- Brandmightbea moresignificantfactor influencingsalespricethanrating.



CorrelationMatrix:

The above image shows a correlation matrix, which is a table showing the correlation coefficientsbetweenmultiplevariables.Inthiscase,thevariablesare:sales_price,rating, no_of_reviews, and left_in_stock.

- **Keypoints:**

  ➢ **Diagonalvalues:**

  Thediagonalvaluesarealways1, asavariablealwaysperfectlycorrelates withitself.

- **Positivecorrelation:**

A positive value indicates that the two variables tend to move in the same direction.Forinstance,the0.09correlationbetween"sales_price"and"rating"suggests that higher-priced items tend to have slightly higher ratings.

- **Negativecorrelation:**

Anegative value indicatesthatthetwovariablestendto move inopposite directions.For example, the -0.15 correlation between "no_of_reviews" and "left_in_stock" implies that items with more reviews tend to have less stock left.

- **Strengthofcorrelation:**

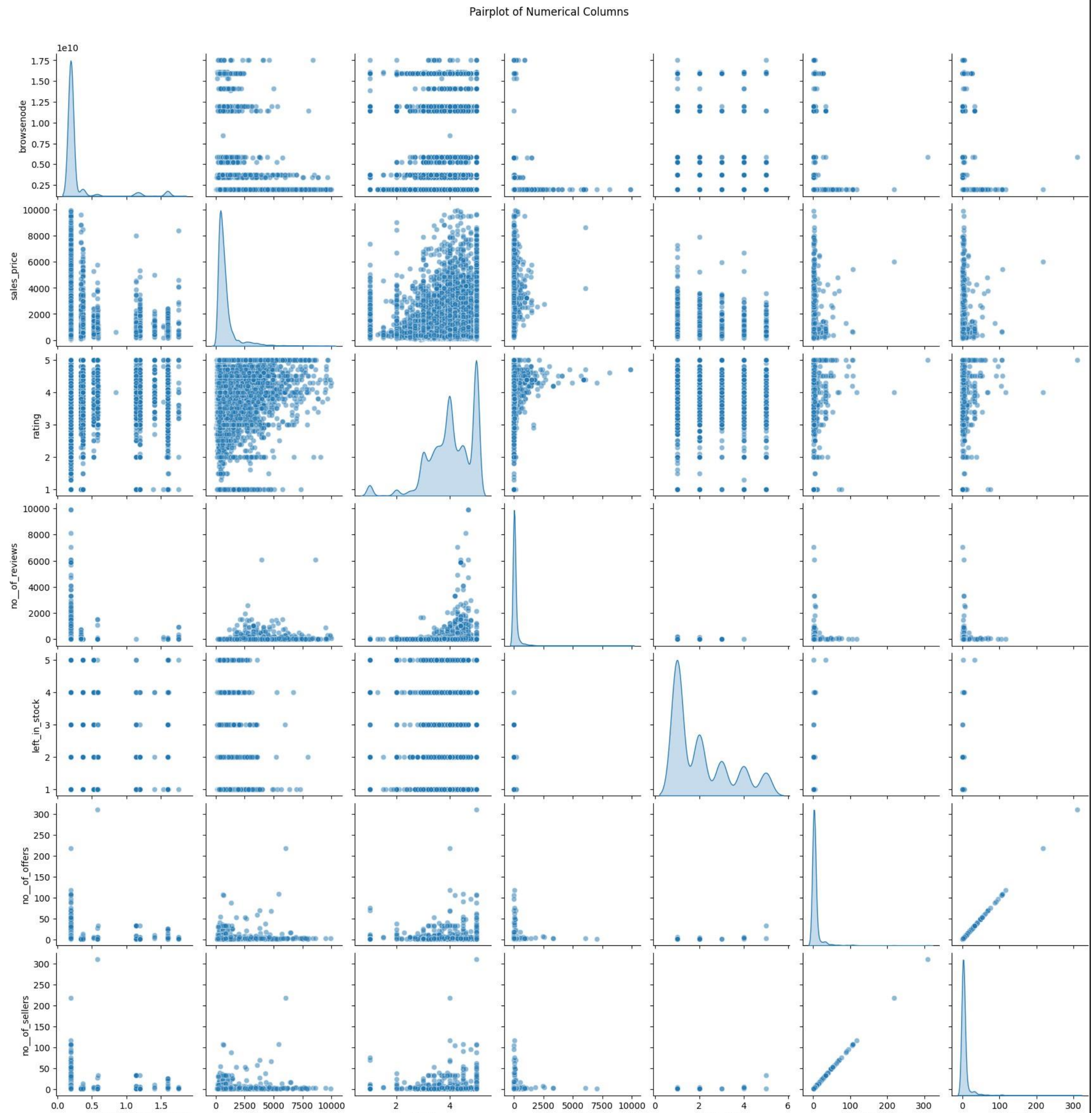
Thecloserthevalue isto 1or-1,thestrongerthecorrelation.Valuescloser to 0 indicate a weaker relationship.

- **Interpretingthematrix:**

Inthisexample, therearenostrongcorrelations.

- Thestrongest positivecorrelationisbetween"sales_price"and"rating"(0.09).
- Thestrongest negativecorrelationisbetween"no_of_reviews"and"left_in_stock"(-0.15).
- Otherrelationshipsareweak,indicatinglittletonolinearassociationbetweenthe variables.

```
#BIVARIATEANALYSISAssuming'data'isyourDataFrame

#1.Numericalvs.Numerical
# Pairplot to visualize pairwise relationships
sns.pairplot(data,diag_kind='kde',plot_kws={'alpha':0.5})
plt.suptitle("Pairplot of Numerical Columns", y=1.02)
plt.show()
```



PairPlot:

The imageshowsapairplot,alsoknownasascatterplotmatrix,whichisa powerfultoolfor data visualization in statistics and machine learning. It helps to understand the relationships between multiple variables in a dataset at a glance.

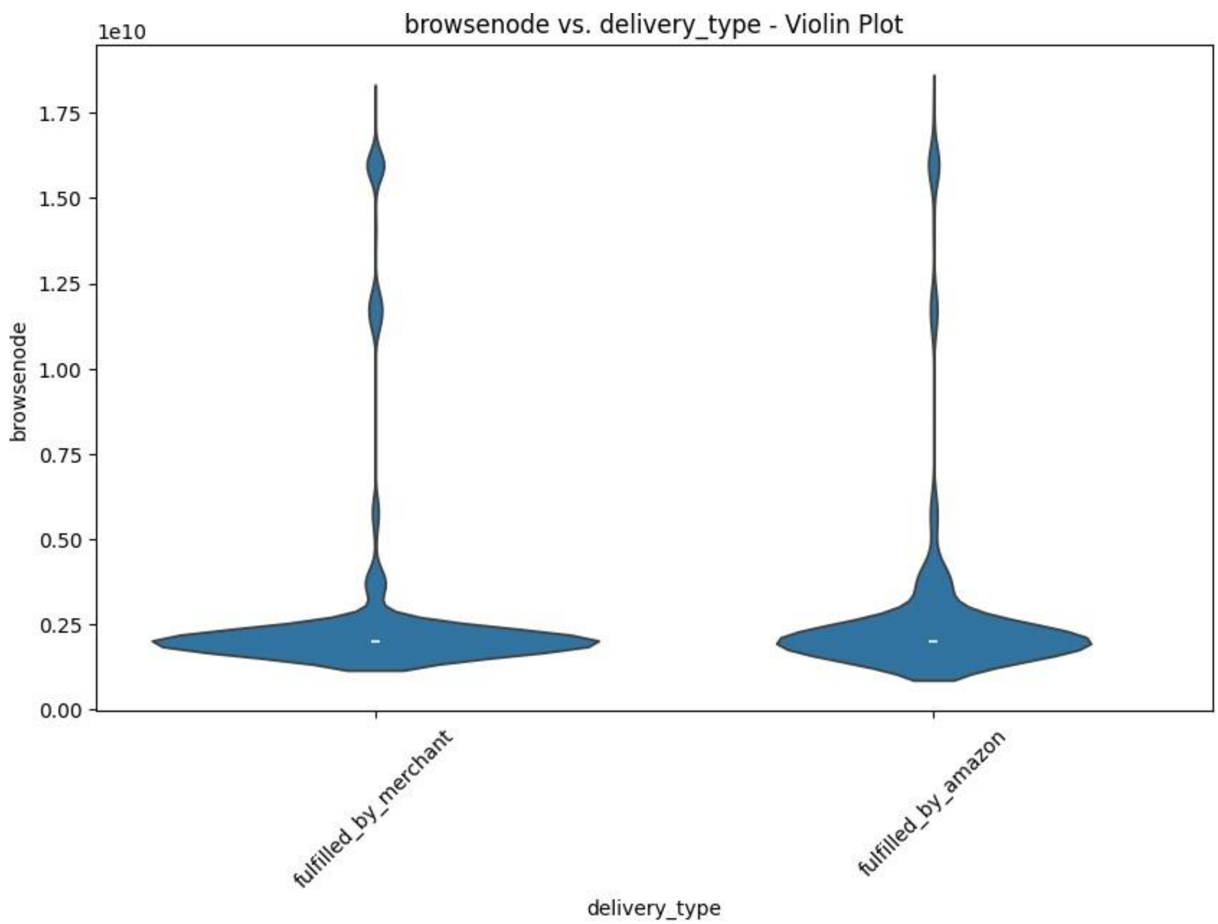
KeyPoints:

- **Diagonal:**

The diagonal cells display histograms for each variable, showing the distribution of theirvalues.Thishelpstounderstandthefrequencyandspreadofeachvariable individually.

- **Off-Diagonal:**

The off-diagonal cells display scatterplots for each pair of variables. Each dot in a scatterplotrepresentsasingledatapoint,withitsx-coordinatecorrespondingtothevalue of one variable and its y-coordinate corresponding to the value of another variable.



DataCleaning:

1.NullHandling

Nullvaluesare missingorundefinedentriesinadataset. Effectivehandling iscrucialfor maintaining the integrity of the analysis.

MethodstoHandleNulls:

- **RemovalofNulls**:Droprowsor columnswithnullvaluesifthemissingdata isminimalor non-essential.
  - df.dropna()toremoverows/columnswithnulls.
- **Imputation:**Replacenullvalueswithmeaningfulsubstituteslike:
  - Mean,median,ormodefornumericaldata.
  - Aplaceholder valueor themostfrequentcategoryforcategoricaldata.
  - Example:df['column'].fillna(df['column'].mean(),inplace=True).
- **Flagging:**Adda newcolumnindicatingrowswithmissingvaluesfor futurereference.

Tools:pandasprovidesutilitieslikeisnull(),notnull(),andfillna().

2.StopWords

Stopwordsarecommonlyusedwords(e.g.,"the","and","is")thatoftendo notadd significant meaning to text analysis.

WhyRemoveStopWords?

- Theyreducenoiseinnaturallanguageprocessing(NLP) tasks.
- Improvecomputationalefficiencyandfocusonmeaningfulterms.

LIBRARIES:

- Usepredefinedlistsfromlibrarieslikenltkorspacy.
- Examplewithnltk:

  python
  Copycode
  ```
  from nltk.corpus import stopwords
  stop_words=set(stopwords.words('english'))
  filtered_text=[wordforwordintext.split()if wordnotinstop_words]
  ```

3.Non-EnglishWords

Non-EnglishwordsinadatasetcaninterferewithNLPtasksfocusedonEnglishlanguagedata.

Detectionand Removal:

- **LanguageDetection**:Uselibrarieslikelangdetectorlangidtoidentifythelanguageoftext.
  - Examplewithlangdetect:

    python
    Copycode
    fromlangdetectimportdetect if
    detect(text) == 'en':

4.Punctuation

Punctuationmarks(e.g., !,?,,)maynot holdsignificant meaninginNLPtasksbut couldbe valuable for sentiment or style analysis.

HandlingPunctuation:

- **Removal**:Strippunctuationtofocusonthecoretext.
  - Examplewithstring.punctuation:

    python
    Copycode
    importstring
    text=text.translate(str.maketrans('','',string.punctuation))

- **Retention**:Retainpunctuationfor taskslikesentiment analysisor wherepunctuationconveys meaning.
- **CustomHandling**:Preservecertainmarks(e.g.,@formentions)basedontheusecase

5.ShortFormtoFull Form

Expandingabbreviationsorcontractionsisessentialforstandardizingtextandimprovingclarity.

Examples:

- Shortforms:can't,won't,idk
- Fullforms:cannot, willnot, Idon'tknow

Approach:

- **RegexReplacement**:Definea dictionaryofshortformsandtheir expansions, thenreplace them using regular expressions.

  python
  Copycode
  contractions={"can't":"cannot","won't":"willnot","idk":"Idon'tknow"} import re

pattern=re.compile(r'\b('+'|'.join(contractions.keys())+r')\b') text = pattern.sub(lambda x: contractions[x.group()], text)

Stemming:

Stemming istheprocessofreducingawordtoitsbaseorrootform, oftenbyremoving suffixes. It focuses on achieving a normalized representation, regardless of whether the resulting form is a valid word.

How ItWorks:

- Reduceswordsto theirstemsbyapplying rule-basedmethods.
- Doesnotconsiderthecontextormeaningofthe word.

Example:

"running"→"run"

"better"→"better"(unchanged,dependingonthestemmer used)

CommonAlgorithms:

- **PorterStemmer:**Awidelyusedrule-basedstemmer.

Python:

```
fromnltk.stemimportPorterStemmer

stemmer = PorterStemmer()

words=["running", "runner","runs"]

stemmed_words=[stemmer.stem(word)forwordinwords] print(stemmed_words)#

Output: ['run', 'runner', 'run']
```

- **SnowballStemmer**: AmoreadvancedversionofthePorterStemmer,supporting multiple languages.

Python:

```
from nltk.stem import SnowballStemmer

stemmer = SnowballStemmer("english")

words=["happily","happiness","happiest"]

stemmed_words = [stemmer.stem(word) for word in words]

print(stemmed_words)#Output:['happili','happi', 'happiest']
```

Lemmatization:

Lemmatizationistheprocessofreducingawordto itsbaseorrootform,knownasa lemma, while considering the word’s context and its part of speech (POS). Unlike stemming, lemmatization ensures that the resulting word is a valid dictionary word.

Howitworks:

- Based ontheDictionary form
- Normalizewordsforconsistentanalysis(e.g.,"running"→"run")
- Itchooseonthe **Frequency**word

Library:

NLTK(NaturalLanguage Toolkit)

ExampleOutput:

```
Product Names (with stopwords removed, lemmatized, and Porter stemming) and Ratings:
                                          product_name rating
0      la facon cotton kalamkari handblock saree blou...    5.0
1          sf jean pantaloon man  plain slim fit t shirt    3.6
2      lovista cotton gota patti tassel traditional p...    3.5
3                  people man  print regular fit t shirt    3.0
4      monte carlo grey solid cotton blend polo colla...    5.0
...                                                  ...    ...
29995          indian virasat woman  rayon anarkali kurta    5.0
29996       urban ranger pantaloon man  slim fit t shirt    3.0
29997             peter england man  regular fit t shirt    4.0
29998  pinky pari woman  embroider short denim straig...    4.0
29999  gutsy man  full sleeve all over print navy t s...    4.0

[30000 rows x 2 columns]
```

The system leverages various natural language processing (NLP) and machine learning techniquestorecommendsimilarproductsbasedontheir names,embeddings,andbrand.

Thereportcoversthefollowing approaches:

1. **TF-IDFVectorizerwithCosineSimilarity**
2. **Word2Vec Embeddings**
3. **TF-IDF-WeightedWord2VecEmbeddings**
4. **Brand-BasedFilteringwithTF-IDF-WeightedWord2Vec**

1. TF-IDFVectorizerwithCosineSimilarity

Theoretical Foundation

TF-IDF(TermFrequency-InverseDocumentFrequency) isastatisticalmethodthat transforms textual data into numerical vectors by:

- Measuringtermimportancewithinindividualdocuments
- Reducingtheweightofcommonlyoccurringwords
- Creatingauniquevectorrepresentationforeachproductdescription

MathematicalFormulation:

KeyFormulas:

1. TermFrequency(TF):TF(t,d)=ft,d∑kfk,d $TF(t,d)=\frac{f_{t,d}}{\sum_k f_{k,d}}$
2. InverseDocumentFrequency(IDF):IDF(t)=log(Ndft) $IDF(t)=\log\left(\frac{N}{df_t}\right)$
3. TF-IDFScore: $TF-IDF(t,d) = TF(t,d) \times IDF(t)$ TF−IDF(t,d)=F(t,d)× DF(t)
4. cosineSimilarity: $\cos(\theta)=\vec{A}\cdot\vec{B}|\vec{A}||\vec{B}|$ $\cos(\theta)=\frac{\vec{A}\cdot\vec{B}}{|\vec{A}||\vec{B}|}$

ImplementationApproach

- Convertproductnamesintonumericalvectors
- Calculatevectorsimilaritiesusingcosinesimilarity
- Recommendproductswithhighestsimilarityscores

Advantages

- **Simpleand interpretable**

- Computationallyefficient
- Workswellwithshorttextdescriptions

Limitations

- Doesnotcapturesemanticrelationships
- Sensitivetoexactwordmatches
- Struggleswithsynonymsand contextualvariations

Together, TF-IDF identifies terms that are important for distinguishing between products. Cosine similarity, which measures the angle between two vectors, is used to compare the productnamesbytheirTF-IDFvectors.Thecloserthecosinevalue isto1,the moresimilar the products are.

KeyCode Snippets

```
# Initialize TfidfVectorizer
vectorizer = TfidfVectorizer()

# Fit and transform the product names
X_tfidf = vectorizer.fit_transform(data['product_name'])

# Function to recommend products
Tabnine | Edit | Test | Explain | Document | Ask
def recommend_products(product_id, num_recommendations, vectors):
    product_index = data[data['uniq_id'] == product_id].index[0]
    distances = pairwise_distances(vectors[product_index], vectors, metric='cosine')
    recommended_indices = distances.argsort()[0][1:num_recommendations + 1]
    return data.iloc[recommended_indices][['uniq_id', 'product_name', 'sales_price', 'rating', 'medium']]
```

Results:

2. <u>Word2Vec Embeddings</u>

TheoreticalFoundation

Word2Vecgeneratesdensevectorrepresentationsthatcapturesemanticrelationshipsbetween words by analyzing their contextual usage.

KeyTechniques

- **ContinuousBagofWords(CBOW):**Predictstargetwordfromcontext
- **Skip-Gram:**Predictscontextwordsfromtargetword

MathematicalRepresentation

ProductEmbeddingCalculation:ProductEmbedding=1n∑i=1nWordEmbeddingi$\text{ProductEmbedding} = \frac{1}{n}\sum_{i=1}^{n} \text{WordEmbedding}_i$

ImplementationStrategy

- Trainembeddingsonlargeproductdescriptioncorpus
- Averagewordembeddingsto createproduct-levelrepresentations
- Computesimilaritiesbetweenproductembeddings

Advantages

- Capturessemanticmeaning
- Handles contextualvariations
- Discoverslatentrelationshipsbetweenwords

Limitations

- Requireslargetrainingcorpus
- Computationallyintensive
- Treatsallwordsequally

a**Overview:** Word2Vec is a powerful modelthat generates vector representations of words based on the context in which theyappear. Byaveraging the embeddings of words in a productname, wecreateaproduct-levelembeddingthat capturesthesemantic meaningofthe product name. This method is especially useful for understanding the context of words, allowing for more context-aware recommendations.

Explanation: Word2Vecworksbyanalyzingthecontextofwordsinlargecorpustogenerate word embeddings that capture semantic relationships between words. In the case of product names, each word's vector is averaged to generate a product-level vector that represents the entire product description.

KeyCode Snippets

```python
# Tokenize product names
data['product_name_tokens'] = data['product_name'].str.split()

# Train Word2Vec model
model = Word2Vec(
    sentences=data['product_name_tokens'],
    vector_size=100,
    window=5,
    min_count=1,
    workers=4,
    sg=1  # Skip-gram model
)

# Compute product embeddings
Tabnine | Edit | Test | Explain | Document | Ask
def get_product_embeddings(tokens, model):
    vectors = [model.wv[word] for word in tokens if word in model.wv]
    return sum(vectors) / len(vectors) if vectors else None

data['embedding'] = data['product_name_tokens'].apply(lambda tokens: get_product_embeddings(tokens, model))
```

Results:

---

3. TF-IDF-WeightedWord2VecEmbeddingsHybridApproach

CombinesstrengthsofTF-IDFandWord2Vecbyweightingwordembeddingsbasedontheir importance.

MathematicalFormula

WeightedEmbedding=∑i=1n(TF−IDFi×WordEmbeddingi)

$$WeightedEmbedding=\sum_{i=1}^{n}(TF-IDF_i \times WordEmbedding_i)$$

ImplementationMethodology

- CalculateTF-IDFscoresforeach word
- Weight Word2Vec embeddingsusingTF-IDFscores
- Createproductembeddingswithweighted averaging

Advantages

- Balancessemanticmeaningwithtermimportance

- Morenuancedrecommendations
- Reducesimpactoflesssignificantwords

Limitations

- Morecomplex implementation
- Requirescarefulparametertuning

a**Overview:** This approach combines the strengths ofTF-IDF and Word2Vec byweighting theembeddingsofwordsbasedontheirTF-IDFscores.Wordsthat aremoreimportanttothe product description (as determined byTF-IDF) will have a greater influence on the resulting product embedding.

Explanation: Bymultiplyingeachword'sWord2VecembeddingbyitscorrespondingTF-IDFscore, this method ensures that the more importantterms have a higher impact onthe product'sfinalembedding.ThisapproachcombinesthesemanticpowerofWord2Vecwith the importance weighting provided byTF-IDF, creating more accurate and meaningful embeddings for product recommendations.

KeyCode Snippets

```
# Train a TF-IDF vectorizer
vectorizer = TfidfVectorizer(tokenizer=lambda x: x, lowercase=False)
tfidf_matrix = vectorizer.fit_transform(data['product_name_tokens'])

# Compute TF-IDF-weighted Word2Vec embeddings
def get_tfidf_weighted_embeddings(tokens, model, tfidf_matrix, idx):
    vectors = []
    tfidf_scores = tfidf_matrix[idx].toarray()[0]
    for word in tokens:
        if word in model.wv and word in vectorizer.vocabulary_:
            weight = tfidf_scores[vectorizer.vocabulary_[word]]
            vectors.append(model.wv[word] * weight)
    return sum(vectors) / len(vectors) if vectors else None

data['embedding'] = [
    get_tfidf_weighted_embeddings(tokens, model, tfidf_matrix, idx)
    for idx, tokens in enumerate(data['product_name_tokens'])
]
```

Results:

4. Brand-BasedFilteringwithTF-IDF-WeightedWord2Vec

CoreConcept

Filtersandrecommendsproductswithinthesame brandecosystemusingadvanced embedding techniques.

Implementation Strategy

- Segmentproductsbybrand
- ApplyTF-IDF-WeightedWord2Vecwithinbrandsubset
- Recommendsimilarproductsfromsamebrand

Advantages

- Maintainsbrandconsistency
- Providesmoretargetedrecommendations
- Usefulforbrand-specificmarketingstrategies

Limitations

- Reducesrecommendationdiversity
- Maymisscross-brandsimilarities

a**Overview**

This method refines recommendations by considering only products from the same brand. TF-IDF-WeightedWord2Vecembeddingsareusedtocomputesimilaritieswithinthebrand.

KeyCode Snippets:

```python
# Function to recommend products from the same brand
def recommend_products_by_brand(product_id, num_recommendations):
    selected_product = data[data['uniq_id'] == product_id]
    selected_brand = selected_product['brand'].values[0]
    same_brand_products = data[data['brand'] == selected_brand]

    if len(same_brand_products) <= num_recommendations:
        return same_brand_products

    selected_embedding = selected_product['embedding'].values[0].reshape(1, -1)
    embeddings = same_brand_products['embedding'].tolist()
    similarities = cosine_similarity(selected_embedding, embeddings).flatten()
    indices = similarities.argsort()[-num_recommendations:][::-1]
    return same_brand_products.iloc[indices]
```

Results:

ComparativeAnalysis:

TF-IDF:

Word2Vec:

TF-IDF-WeightedWord2Vec:

Brand-BasedFiltering alongwithtfidfand word2vec:

Technique	Semantic Understanding	Computational Complexity	Recommendation Precision
TF-IDF	Low	Low	Moderate
Word2Vec	High	High	Good
TF-IDF-Weighted Word2Vec	VeryHigh	VeryHigh	Excellent
Brand-Based Filtering	VeryHigh	moderate	Brand-Specific

DetailedComparativeAnalysisofRecommendationModels:

Model	Visual Observations (Image-Specific)	Strengths	Weaknesses
TF-IDFVectorizer	Highly inaccurate recommendations. - Includes products with vastly different patterns,styles, and even genders. - Noalignmentto the plaid check pattern or color scheme of the query product. - Example: Dressesorshirts with solid colors and styles are recommended, missingtheplaid pattern entirely.	Easytoimplement and computationally efficient. Uses term frequency to match basic productmetadata (e.g., tags).	Fails to understand visual/contextual features like patterns, colors, and fabric style. Recommends itemsoutsidethe intendedproduct category or gender.
Word2Vec	Recommendations arecloserinstyleto thequeryproduct.	Excelsat capturing semanticsimilarity	Fails to consistently capture specific

	- Focuses on solid ordarkercolors. - Misses the plaid pattern or the red/white/blue color scheme of the query. - Example: Dark solid shirts and light-colored plain shirtsareincluded, whichlack plaidalignment.	betweenproducts. Avoids unrelated recommendations.	featureslikecolor combinations or intricate plaid patterns. May prioritize overall similarity overexactdetails.
TF-IDF-Weighted Word2Vec	Shows much bettercontextual understanding. - Matches plaid pattern but not exact red/white/blue colorcombination. - Example:A green plaid shirt appears,matching the pattern style but deviating from the query's color scheme.	Combines global contextual relevance with local keyword importance, leading to more accurate recommendations. Balances styleand relevance.	Computationally more intensive. Still lacks perfect color matching, important for aesthetic-focused users.
Brand-Based Filtering	Visually aligned with the brand’s designlanguage. - Focuseson design and stylistic consistency. - Mayignoreplaid patterns or specific color preferences if absent in the brand catalog. - Example:	Ensures consistency with thebrandidentity, appealing to brand-loyal customers.	Limited by the diversity of the brand catalog. May ignore stylisticelements outside the brand's focus.

	Recommends shirts with solid colors, stripes, or partial plaid, focusingonbrand style over query-product features.		

CollaborativeFiltering

Collaborative Filtering (CF) is a popular recommendation systemtechnique that makes predictions about a user's preferences based on their previous interactions and the preferences of other users. The fundamental assumptionofcollaborative filtering isthat userswho haveagreedonpast preferencesare likelyto agreeon future preferences. This approach is widely used in applications like e-commerce, streaming services, and social media.

TypesofCollaborativeFiltering

1. **User-BasedCollaborativeFiltering**
   - Thismethodfindssimilarities betweenusersbasedontheirbehavioror preferences.
   - Forexample, ifUserAandUserBhaveratedseveralitemssimilarly,UserA'spreferences can predict User B's preferences for items they haven't rated yet.
   - **Steps**:
     1. Calculatethesimilaritybetweenusersusingmetricslikecosinesimilarity,Pearson correlation, or Jaccard index.
     2. Usethesesimilaritiestogenerate recommendations.
2. **Item-BasedCollaborativeFiltering**
   - Thismethodfocusesonsimilaritiesbetweenitemsinsteadofusers.
   - Forexample, iftwo itemsare frequentlyratedorboughttogether,one itemcanrecommend the other.
   - **Steps**:
     1. Calculatesimilaritybetweenitemsbasedonuserinteractionpatterns.
     2. Recommend itemssimilar tothosetheuser hasalreadyinteracted with.

KeyTechniques

1. **Memory-BasedCollaborativeFiltering**
   - Usesstatisticalmethodsdirectlyonthe data.
   - Simplerandeasiertoimplement.
   - Workswellforsmalldatasetsbutsuffers fromscalabilityissuesasthedataset grows.
2. **Model-BasedCollaborativeFiltering**
   - Employs machine learning models like matrix factorization (e.g., Singular Value Decomposition,AlternatingLeastSquares)todiscoverlatent factorsthatinfluenceuser-item interactions.
   - Scalableandeffectiveforlargedatasets.

AdvantagesofCollaborative Filtering

1. **PersonalizedRecommendations**:Generatestailoredsuggestionsbasedonuserpreferencesand behaviors.
2. **NoDomainKnowledgeRequired**:CFreliessolelyonuser-iteminteractiondatawithout needing domain-specific information about items.
3. **DynamicAdaptation**:Updatesrecommendationsasnewdatabecomes available.

Challenges

1. **Cold-StartProblem**:Difficult torecommend fornewusersornew itemsdueto alackofinteraction data.
2. **DataSparsity**:Insystemswithlargedatasets,most usersinteractwithasmallsubset ofitems, leading to sparse user-item matrices.
3. **Scalability**:Computationalcostincreaseswiththenumberofusersanditemsinthe system.

Applications

- **E-commerce**:Suggestingproductsbasedonpurchasehistory.
- **StreamingServices**:Recommendingmoviesorsongsbasedonviewingorlistening history.
- **SocialMedia**:Proposingfriends,groups,orpostsbasedoninteractions.

```python
from tensorflow.keras.applications import VGG16
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.vgg16 import preprocess_input
from tensorflow.keras.layers import GlobalMaxPooling2D
from tensorflow import keras
import numpy as np
import pandas as pd
import os

# Image processing and model setup
img_width, img_height, chnl = 224, 224, 3  # Set image dimensions
vgg = VGG16(include_top=False, weights='imagenet', input_shape=(img_width, img_height, chnl))
vgg.trainable = False
model1 = keras.Sequential([vgg, GlobalMaxPooling2D()])  # Use keras from tensorflow
model1.summary()

# Function to create image path
def img_path(img_name):
    return os.path.join("/content/ai_stylist_data/images", img_name)

# Function to get embeddings from the model
def model_predict(model, img_name):
    img = image.load_img(img_path(img_name), target_size=(img_width, img_height))  # Load image
    x = image.img_to_array(img)  # Convert image to array
    x = np.expand_dims(x, axis=0)  # Expand dimensions to match model input
    x = preprocess_input(x)  # Preprocess image input
    return model.predict(x).reshape(-1)  # Predict and reshape output

# Path to the embeddings file
embeddings_file_path = '/content/ai_stylist_data/embeddings_subset.csv'
# Check if the embeddings file already exists
if not os.path.exists(embeddings_file_path):
    print("Embeddings file not found, generating embeddings...")
```

```python
        # Generate embeddings for the subset of images
        df_embedding_subset = df_copy_subset['imagePath'].apply(lambda x: model_predict(model1, x))  # Apply model to each image
        # Convert the embeddings into a DataFrame
        df_embedding_subset = df_embedding_subset.apply(pd.Series)
        # Add embeddings to the copied subset DataFrame
        df_copy_subset = pd.concat([df_copy_subset, df_embedding_subset], axis=1)
      # Save the resulting DataFrame with embeddings for the first 2,000 images
        df_copy_subset.to_csv(embeddings_file_path, index=False)
    # Display the first few rows of the resulting DataFrame
        print(df_copy_subset.head())
else:
        print("Embeddings file already exists, skipping the embedding generation.")
```

Output:

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_
58889256/58889256 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Model: "sequential"
```

Layer (type)	Output Shape	Param #
vgg16 (Functional)	(None, 7, 7, 512)	14,714,688
global_max_pooling2d (GlobalMaxPooling2D)	(None, 512)	0

```
 Total params: 14,714,688 (56.13 MB)
 Trainable params: 0 (0.00 B)
 Non-trainable params: 14,714,688 (56.13 MB)
Embeddings file not found, generating embeddings...
1/1 ━━━━━━━━━━━━━━━━━━━━ 3s 3s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
```

```python
# All columns
merged_df.head()
```

	id	gender	masterCategory	subCategory	articleType	baseColour	season	year	usage	productDisplayName	...	502	503	504	505	506	507	508	509	510	511
0	15970	Men	Apparel	Topwear	Shirts	Navy Blue	Fall	2011.0	Casual	Turtle Check Men Navy Blue Shirt	...	0.000000	23.615625	0.000000	6.146007	0.000000	25.227303	18.948944	0.000000	6.717718	0.000000
1	39386	Men	Apparel	Bottomwear	Jeans	Blue	Summer	2012.0	Casual	Peter England Men Party Blue Jeans	...	0.000000	94.777770	0.000000	21.044895	0.000000	28.778320	35.437466	0.000000	40.000677	7.511488
2	59263	Women	Accessories	Watches	Watches	Silver	Winter	2016.0	Casual	Titan Women Silver Watch	...	0.084442	0.000000	1.313956	0.000000	0.000000	25.465756	9.873016	0.000000	11.206230	0.000000
3	21379	Men	Apparel	Bottomwear	Track Pants	Black	Fall	2011.0	Casual	Manchester United Men Solid Black Track Pants	...	0.000000	137.025470	0.000000	9.216299	19.427944	14.087112	46.465460	1.490025	5.397563	6.753218
4	53759	Men	Apparel	Topwear	Tshirts	Grey	Summer	2012.0	Casual	Puma Men Grey T-shirt	...	0.000000	34.992783	0.000000	3.899958	0.000000	18.679634	85.453600	0.000000	53.367590	0.000000

5 rows × 520 columns

OverviewofVGG16Embedding-BasedModel

TheVGG16 embedding-based model leverages theVGG16 architecture, a popular convolutional neural network(CNN),pre-trainedontheImageNet dataset,toextract richfeaturerepresentationsfromimages. These embeddings can serve as input for various tasks, such as content-based recommendation systems, image classification, clustering, or retrieval.

VGG16Architecture

1. **Developedby**:VisualGeometryGroup(VGG) atOxfordUniversity.
2. **KeyFeatures**:
   - Consistsof16 layers(13 convolutionallayersand3 fullyconnectedlayers).
   - Usessmallconvolutionfilters(3x3)withstride1.
   - Incorporatesmaxpoolingfor down-sampling.
   - Employsafixed inputsizeof224x224pixelsforimages.
3. **Pre-TrainedWeights**:TrainedonImageNet,alarge-scaledatasetcontainingmillionsoflabeled images.

EmbeddingExtractionwithVGG16

1. **FeatureExtraction**:
   - Removethefullyconnectedlayers(classificationhead).
   - Usetheoutputofthelastconvolutionallayerorthepenultimatelayerasthe embedding.
   - Theseembeddingscapturehigh-levelvisualfeaturessuchasshapes, textures, andpatterns.
2. **Process**:
   - LoadtheVGG16modelpre-trainedonImageNet.
   - Preprocessimages(resizeto224x224,normalizepixelvalues, etc.).
   - Passeach imagethroughthemodelto obtainembeddings.
3. **Storage**:
   - Saveembeddingsinastructuredformat (e.g.,NumPyarrays,database,orfilestorage)for future use.

Applications

1. **Content-BasedRecommendationSystems**:
   - Findvisuallysimilaritems, suchasclothing, artwork,or furniture.
   - Usesimilaritymeasures(e.g.,cosinesimilarity,Euclideandistance)tocompareembeddings.
2. **ImageRetrieval**:

- Searchfor images inadatabasethataresimilar toagivenqueryimage.

- Usefulinapplicationslikestock photolibrariesorvisualsearchengines.

3. **ImageClustering**:
    - Groupimageswithsimilarvisualfeaturesforcategorizationorpatterndiscovery.
    - Appliedindomainslikee-commerceandmedicalimaging.
4. **HybridRecommendation Systems**:
    - Combineembeddingswithmetadata(e.g.,productdescriptions,ratings)forenhanced recommendations.

Advantages ofVGG16Embeddings

1. **Pre-TrainedModel**:
    - Leveragestheknowledge fromImageNet,reducingtheneed forextensivetrainingon specific datasets.
2. **High-LevelFeatures**:
    - Capturescomplexvisualpatternsandrepresentationsfromimages.
3. **Versatility**:
    - Suitableforvariousdownstreamtasksbeyond classification.

CODE

```python
# Function to recommend products based on embeddings
def recommend_products(input_img_name, top_n=5):
    # Get input image embedding
    input_embedding = model_predict(model1, input_img_name).reshape(1, -1)

    # Compute cosine similarity
    similarities = cosine_similarity(input_embedding, embeddings).flatten()

    # Get top N recommendations (excluding the input image itself if it's in the dataset)
    recommended_indices = np.argsort(similarities)[::-1][:top_n]
    recommended_scores = similarities[recommended_indices]

    # Display recommendations
    input_img_path = img_path(input_img_name)
    display_recommendations(input_img_path, recommended_indices, recommended_scores)

# Example usage
input_image_name = '30039.jpg'  # Replace with your image filename
recommend_products(input_image_name, top_n=5)
```

```python
# Extract embeddings as a numpy array
embeddings = df.iloc[:, -512:].values  # Assuming embeddings are the last 512 columns
# Function to display images with product names and similarity scores
def display_recommendations(input_img_path, recommended_indices, scores):
    fig, axes = plt.subplots(1, len(recommended_indices) + 1, figsize=(15, 5))

    # Display input image
    img = image.load_img(input_img_path, target_size=(224, 224))
    axes[0].imshow(img)
    axes[0].set_title("Input Image")
    axes[0].axis('off')
    # Display recommended images
    for i, idx in enumerate(recommended_indices):
        rec_img_path = img_path(df.iloc[idx]['imagePath'])  # Get image path
        rec_img = image.load_img(rec_img_path, target_size=(224, 224))
        axes[i + 1].imshow(rec_img)
        axes[i + 1].set_title(f"{df.iloc[idx]['productDisplayName']}\nScore: {scores[i]:.2f}")
        axes[i + 1].axis('off')

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

```python
# Example usage
input_image_name = '30039.jpg'  # Replace with your image filename
recommend_products(input_image_name, top_n=5)
```

Output:

RecommendationSystemUsingResNet50Embeddings Overview
Arecommendation system leveraging ResNet50 embeddings is a hybrid approach that combines deep learning and traditionalrecommendationtechniques. ResNet50, apopular convolutionalneuralnetwork (CNN),ispre-trainedontheImageNet datasetandiswidelyused forfeatureextractionfromimages.By generating embeddings (numerical representations) for images, ResNet50 captures high-level visual features, which can be utilized to recommend similar items based on image content.

WorkflowfortheSystem

1. **DatasetPreparation**
   - Collect adatasetofimagesassociatedwiththe itemstoberecommended(e.g.,product images for an e-commerce platform).
   - IncludemetadatalikeitemIDs,names, prices, ordescriptions.
2. **FeatureExtractionUsing ResNet50**
   - Loadthe ResNet50modelpre-trainedonImageNet.
   - Removetheclassificationheadtoobtainthefeatureextractormodel.
   - Passeach imagethroughthemodeltoextract embeddingsfromthepenultimatelayer.
   - Storetheseembeddings inadatabaseforfurther processing.
3. **SimilarityComputation**
   - Calculatethesimilaritybetweenembeddingstofindvisuallysimilaritems.
   - Usesimilaritymetricslike:
     - **CosineSimilarity**:Measuresthecosineoftheanglebetweentwoembeddingvectors.
     - **EuclideanDistance**:Measuresthestraight-linedistancebetweentwo pointsinthe embedding space.
   - Createa similaritymatrixwhere each itemiscomparedwitheveryotheritem.

4. **RecommendationGeneration**
   - Foragivenitem,findthetopNsimilaritems basedonsimilarityscores.
   - Displaytheserecommendationsalongwithrelevantmetadata(e.g.,productname, price,or rating).
5. **IntegrationwithMetadata**
   - Enhancerecommendationsbycombining image-based featureswithother metadata,suchas textual descriptions or user behavior.
   - Usehybridapproachestoimprovethesystem'sperformance.

AdvantagesofUsingResNet50

1. **Pre-TrainedExpertise**:ResNet50'spre-trainedweightsonImageNetproviderobustfeature extraction capabilities, reducing the need for extensive training.
2. **VisualRecommendations**:Usefulfordomainswherevisualappearancesignificantlyinfluences user preferences (e.g., fashion, furniture, art).
3. **Scalability**:Embeddingsallowforefficientsimilaritycomputationsandscalablerecommendation generation.

Challenges

1. **ColdStartforNon-VisualItems**:Thesystemmaynot performwellfor itemswithout meaningful visual content.
2. **ComputationalCost**:Featureextractionandsimilaritycomputationcanberesource-intensive for large datasets.
3. **LimitedContextUnderstanding**:Purelyvisualembeddingsmight misscontextualrelevance provided by metadata or user behavior.

Applications

- **E-commerce**:Recommendingvisuallysimilarproducts,suchasclothingoraccessories.
- **ArtPlatforms**:Suggestingartworkor designsbasedonvisualstyle.
- **FurnitureStores**:Helpingusersfind furniturepieceswithsimilar aesthetics.

```python
# Example recommendation function
def recommend_similar_images(image_id, top_n=5):
    # Find the embedding of the query image
    query_embedding = df[df['id'] == image_id]['embeddings'].values[0]

    # Compute cosine similarity
    all_embeddings = np.stack(df['embeddings'].values)
    similarity_scores = cosine_similarity([query_embedding], all_embeddings).flatten()
    # Get top N similar images
    top_indices = similarity_scores.argsort()[-top_n-1:][::-1][1:]  # Exclude the query image itself
    similar_images = df.iloc[top_indices]

    # Display the images
    plt.figure(figsize=(15, 5))
    for i, (_, row) in enumerate(similar_images.iterrows()):
        img = image.load_img(row['fullImagePath'], target_size=(224, 224))
        plt.subplot(1, top_n, i + 1)
        plt.imshow(img)
        plt.axis('off')
        plt.title(f"Similarity: {similarity_scores[top_indices[i]]:.2f}")
    plt.show()
```

```python
# Test the recommendation function
test_image_id = 57958    # Replace with an actual ID from the styles.csv
recommend_similar_images(test_image_id, top_n=5)
```

Output: