Project 1: Customer Sentiment Analysis for E-commerce Reviews

Objective

Analyze a dataset of customer reviews from an e-commerce platform to predict sentiment (positive/negative/neutral) and identify key factors contributing to negative reviews.

Assignment Tasks

1. Data Loading and Initial Inspection

1. Load the dataset into a Pandas DataFrame.
2. Print the column names, data types, and descriptive statistics.
3. Check for missing values and duplicates in the dataset.
4. Summarize the distribution of review sentiments (target variable).

Deliverables:

* Overview of the dataset.
* Insights on missing values, duplicates, and sentiment distribution.

2. Data Cleaning and Preprocessing

1. Handle missing values:
   * Impute numerical columns with mean or median.
   * Drop rows with missing review text.
2. Remove duplicate rows, if any.
3. Text preprocessing:
   * Remove stopwords, punctuation, and numbers.
   * Convert text to lowercase.

Deliverables:

* Cleaned dataset with preprocessed review text.

3. Exploratory Data Analysis (EDA)

1. Plot the distribution of review lengths.
2. Create a word cloud for each sentiment category.
3. Visualize the frequency of top words in negative reviews.

Deliverables:

* Word clouds and frequency visualizations.
* Insights into review characteristics.

4. Feature Engineering

1. Create numerical features:
   * Review length (word count).
   * Sentiment polarity (using TextBlob).
2. Perform TF-IDF vectorization on review text.

Deliverables:

* TF-IDF matrix for text data.
* Numerical features for further analysis.

5. Correlation and Feature Selection

1. Compute correlation for numerical features.
2. Select important features (e.g., review length, polarity).

Deliverables:

* Heatmap of correlations.
* Final feature set for modeling.

6. Model Building

1. Split the dataset into training (80%) and testing (20%) sets.
2. Train the following models:
   * Logistic Regression
   * Random Forest
   * Support Vector Machine (SVM)
3. Evaluate models using:
   * Accuracy
   * Precision
   * Recall
   * F1 Score

Deliverables:

* Trained models.
* Performance metrics for comparison.

7. Model Optimization

1. Perform hyperparameter tuning for the best model using GridSearchCV.
2. Evaluate the optimized model on the testing set.

Deliverables:

* Tuned model.
* Improved performance metrics.

8. Insights and Recommendations

1. Identify key factors contributing to negative reviews.
2. Provide actionable recommendations for improving customer satisfaction.

Deliverables:

* Report summarizing insights and recommendations.
* Supporting visualizations.

9. Advanced Analytics (Optional)

1. Use SHAP or LIME for model interpretability.
2. Highlight critical features affecting individual predictions.

Deliverables:

* SHAP or LIME visualizations.

Code for Project 1

The code will include:

* Data loading, preprocessing, and TF-IDF vectorization.
* EDA using matplotlib, seaborn, and wordcloud.
* Model training and hyperparameter tuning.

Project 2: Predicting House Prices

Objective

Analyze a real estate dataset to predict house prices based on features such as location, size, and amenities. The project emphasizes feature engineering, data transformations, and advanced modeling techniques.

Assignment Tasks

1. Data Loading and Initial Inspection

1. Load the real estate dataset into a Pandas DataFrame.
2. Print column names, data types, and descriptive statistics.
3. Check for missing values and duplicates.
4. Summarize the target variable (Price) distribution.

Deliverables:

* Overview of the dataset.
* Insights on missing values and duplicates.

2. Data Cleaning and Preprocessing

1. Handle missing values:
   * Impute numerical columns with median.
   * Impute categorical columns with mode.
2. Remove duplicates.
3. Convert categorical features (e.g., neighborhood) into numerical using label encoding.

Deliverables:

* Cleaned dataset with imputed values.

3. Exploratory Data Analysis (EDA)

1. Plot the distribution of numerical features (e.g., size, number of rooms).
2. Visualize the relationship between house price and location.
3. Use pair plots to analyze feature interactions.

Deliverables:

* Visualizations and insights on house price factors.

4. Feature Engineering

1. Create new features:
   * Price per square foot.
   * Age of the house (current year - year built).
2. Normalize numerical features using Min-Max Scaling.

Deliverables:

* Dataset with engineered features.

5. Correlation and Feature Selection

1. Compute correlation for numerical features.
2. Use feature importance from Random Forest to select key features.

Deliverables:

* Heatmap and final feature set for modeling.

6. Model Building

1. Split the dataset into training (80%) and testing (20%) sets.
2. Train the following models:
   * Linear Regression
   * Random Forest
   * Gradient Boosting (XGBoost)
3. Evaluate models using:
   * RMSE
   * MAE
   * R^2 Score

Deliverables:

* Trained models.
* Performance metrics for comparison.

7. Model Optimization

1. Perform hyperparameter tuning using RandomizedSearchCV.
2. Evaluate the tuned model on the testing set.

Deliverables:

* Optimized model and metrics.

8. Insights and Recommendations

1. Identify key factors driving house prices.
2. Write a report suggesting strategies for buyers/sellers based on findings.

Deliverables:

* Report summarizing key insights.

9. Advanced Analytics (Optional)

1. Use SHAP to interpret the model.
2. Highlight features influencing individual predictions.

Deliverables:

* SHAP visualizations.

Code for Project 2

The code will include:

* Data preprocessing and normalization.
* EDA using matplotlib and seaborn.
* Model training and evaluation using sklearn and xgboost.

Both projects are designed to provide robust learning experiences and actionable insights with complete datasets and coding implementations.

Great choice! The Customer Sentiment Analysis for E-commerce Reviews project will give you hands-on experience with text data preprocessing, feature engineering, and predictive modeling. Here's a step-by-step implementation outline with the corresponding code for the first few tasks.

Step 1: Data Loading and Initial Inspection

1. Load the dataset into a Pandas DataFrame.
2. Print the column names, data types, and descriptive statistics.
3. Check for missing values and duplicates.
4. Summarize the distribution of review sentiments (target variable).

Code Implementation

# Import necessary libraries

import pandas as pd

import numpy as np

# Load the dataset

# Assuming the dataset is in a CSV file named 'ecommerce\_reviews.csv'

df = pd.read\_csv("ecommerce\_reviews.csv")

# Display basic information about the dataset

print("Dataset Info:")

print(df.info())

# Display descriptive statistics

print("\nDescriptive Statistics:")

print(df.describe(include='all'))

# Check for missing values

print("\nMissing Values:")

print(df.isnull().sum())

# Check for duplicate rows

print("\nDuplicate Rows:")

print(df.duplicated().sum())

# Summarize the target variable (Sentiment)

print("\nSentiment Distribution:")

print(df['sentiment'].value\_counts())

Step 2: Data Cleaning and Preprocessing

1. Handle missing values:
   * Impute numerical columns with the median or mean.
   * Drop rows with missing review text.
2. Remove duplicate rows.
3. Text preprocessing:
   * Remove stopwords, punctuation, and numbers.
   * Convert text to lowercase.

Project I Coding

Code Implementation

import string

from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize

# Ensure you have NLTK stopwords downloaded

import nltk

nltk.download('stopwords')

nltk.download('punkt')

# Fill missing values

df['numerical\_column'] = df['numerical\_column'].fillna(df['numerical\_column'].median())

df['review\_text'].dropna(inplace=True)

# Remove duplicates

df = df.drop\_duplicates()

# Define a text preprocessing function

def preprocess\_text(text):

stop\_words = set(stopwords.words('english'))

text = text.lower() # Convert to lowercase

text = ''.join([char for char in text if char not in string.punctuation]) # Remove punctuation

text = ''.join([char for char in text if not char.isdigit()]) # Remove numbers

words = word\_tokenize(text) # Tokenize

words = [word for word in words if word not in stop\_words] # Remove stopwords

return ' '.join(words)

# Apply text preprocessing

df['processed\_text'] = df['review\_text'].apply(preprocess\_text)

# Save the cleaned dataset

df.to\_csv("cleaned\_reviews.csv", index=False)

Step 3: Exploratory Data Analysis (EDA)

1. Plot the distribution of review lengths.
2. Create a word cloud for each sentiment category.
3. Visualize the frequency of top words in negative reviews.

Code Implementation

import matplotlib.pyplot as plt

from wordcloud import WordCloud

# Distribution of review lengths

df['review\_length'] = df['processed\_text'].apply(lambda x: len(x.split()))

plt.hist(df['review\_length'], bins=30, color='skyblue', edgecolor='black')

plt.title("Distribution of Review Lengths")

plt.xlabel("Number of Words")

plt.ylabel("Frequency")

plt.show()

# Generate word clouds for each sentiment

sentiments = df['sentiment'].unique()

for sentiment in sentiments:

text = ' '.join(df[df['sentiment'] == sentiment]['processed\_text'])

wordcloud = WordCloud(width=800, height=400, background\_color='white').generate(text)

plt.figure(figsize=(10, 5))

plt.imshow(wordcloud, interpolation='bilinear')

plt.axis('off')

plt.title(f"Word Cloud for {sentiment} Reviews")

plt.show()

# Frequency plot of top words in negative reviews

from collections import Counter

negative\_reviews = ' '.join(df[df['sentiment'] == 'negative']['processed\_text'])

negative\_words = negative\_reviews.split()

word\_counts = Counter(negative\_words)

most\_common = word\_counts.most\_common(20)

words, counts = zip(\*most\_common)

plt.bar(words, counts, color='red')

plt.xticks(rotation=45)

plt.title("Top Words in Negative Reviews")

plt.ylabel("Frequency")

plt.show()

Next Steps

* Step 4: Perform Feature Engineering by creating numerical features like review length and sentiment polarity.
* Step 5: Split the dataset and build machine learning models (e.g., Logistic Regression, Random Forest, and SVM).
* Step 6: Evaluate models and perform hyper parameter tuning.
* Step 7: Write a final report summarizing the insights.

Project II

Below is a detailed implementation of the project with explanations for each step. This guide includes the necessary Python code for every task, along with instructions for running it.

Ensure you have the required Python libraries installed: pandas, numpy, matplotlib, seaborn, scikit-learn, xgboost, and shap. You can install them using the command:

pip install pandas numpy matplotlib seaborn scikit-learn xgboost shap

**Data Loading and Initial Inspection**

pip install pandas numpy matplotlib seaborn scikit-learn xgboost shap

* 1. **Data Loading and Initial Inspection**

import pandas as pd

# Load the dataset

df = pd.read\_csv('real\_estate\_data.csv') # Replace with your dataset file name

# Overview of the dataset

print(df.head())

print(df.info())

print(df.describe())

# Check for missing values and duplicates

print(df.isnull().sum())

print(f"Number of duplicates: {df.duplicated().sum()}")

# Summarize the target variable

import matplotlib.pyplot as plt

import seaborn as sns

sns.histplot(df['Price'], kde=True)

plt.title('Price Distribution')

plt.show()

Data Cleaning and Preprocessing

# Impute missing values

df.fillna({'column\_name': df['column\_name'].median() for column\_name in df.select\_dtypes(include=['float64', 'int64']).columns}, inplace=True)

df.fillna({'column\_name': df['column\_name'].mode()[0] for column\_name in df.select\_dtypes(include=['object']).columns}, inplace=True)

# Remove duplicates

df.drop\_duplicates(inplace=True)

# Label encode categorical variables

from sklearn.preprocessing import LabelEncoder

categorical\_cols = df.select\_dtypes(include=['object']).columns

le = LabelEncoder()

for col in categorical\_cols:

df[col] = le.fit\_transform(df[col])

print("Data after preprocessing:")

print(df.head())

1. **Exploratory Data Analysis (EDA)**

# Plot distributions

for col in df.select\_dtypes(include=['float64', 'int64']).columns:

sns.histplot(df[col], kde=True)

plt.title(f'{col} Distribution')

plt.show()

# Visualize relationships

sns.scatterplot(x='Location', y='Price', data=df)

plt.title('Price vs Location')

plt.show()

# Pair plot for feature interactions

sns.pairplot(df)

plt.show()

1. **Feature Engineering**

# Create new features

df['Price\_per\_sqft'] = df['Price'] / df['Size']

df['Age'] = 2024 - df['YearBuilt']

# Normalize numerical features

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

numerical\_cols = df.select\_dtypes(include=['float64', 'int64']).columns

df[numerical\_cols] = scaler.fit\_transform(df[numerical\_cols])

print("Dataset after feature engineering:")

print(df.head())

1. **Correlation and Feature Selection**

# Compute correlation

sns.heatmap(df.corr(), annot=True, cmap='coolwarm')

plt.title('Feature Correlation')

plt.show()

# Feature importance using Random Forest

from sklearn.ensemble import RandomForestRegressor

X = df.drop('Price', axis=1)

y = df['Price']

rf = RandomForestRegressor()

rf.fit(X, y)

importances = rf.feature\_importances\_

feature\_importance = pd.DataFrame({'Feature': X.columns, 'Importance': importances})

print(feature\_importance.sort\_values(by='Importance', ascending=False))

1. **Model Building**

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error, r2\_score

from xgboost import XGBRegressor

from sklearn.linear\_model import LinearRegression

from sklearn.ensemble import RandomForestRegressor

# Split data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Train models

models = {

"Linear Regression": LinearRegression(),

"Random Forest": RandomForestRegressor(),

"XGBoost": XGBRegressor()

}

results = {}

for name, model in models.items():

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

results[name] = {

'RMSE': mean\_squared\_error(y\_test, y\_pred, squared=False),

'MAE': mean\_absolute\_error(y\_test, y\_pred),

'R2': r2\_score(y\_test, y\_pred)

}

print(pd.DataFrame(results))

1. **Model Optimization**

from sklearn.model\_selection import RandomizedSearchCV

# Hyperparameter tuning for Random Forest as an example

param\_grid = {

'n\_estimators': [100, 200, 300],

'max\_depth': [None, 10, 20],

'min\_samples\_split': [2, 5, 10],

'min\_samples\_leaf': [1, 2, 4]

}

grid\_search = RandomizedSearchCV(RandomForestRegressor(), param\_grid, cv=3, scoring='neg\_mean\_absolute\_error')

grid\_search.fit(X\_train, y\_train)

best\_model = grid\_search.best\_estimator\_

y\_pred = best\_model.predict(X\_test)

print("Optimized Model Metrics:")

print(f"RMSE: {mean\_squared\_error(y\_test, y\_pred, squared=False)}")

print(f"MAE: {mean\_absolute\_error(y\_test, y\_pred)}")

print(f"R^2: {r2\_score(y\_test, y\_pred)}")

1. **Insights and Recommendations**

# Generate insights

feature\_importance.sort\_values(by='Importance', ascending=False).plot(kind='bar')

plt.title('Feature Importance')

plt.show()

1. **Advanced Analytics**

import shap

# SHAP analysis for the best model

explainer = shap.Explainer(best\_model, X\_test)

shap\_values = explainer(X\_test)

shap.summary\_plot(shap\_values, X\_test)

shap.plots.waterfall(shap\_values[0])

Notes:

1. Replace real\_estate\_data.csv with your dataset file path.
2. Save all results (metrics, plots) for your report.
3. Use Jupyter Notebook for interactive exploration if needed.