```
import pandas as pd
import numpy as np
from textblob import TextBlob
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.optimizers import Adam
import matplotlib.pyplot as plt
```

```
df = pd.read_csv("/content/eCommerce_Customer_support_data.csv")

print("🔹 Dataset Shape:", df.shape)
print("\n🔹 Data Info:")
print(df.info())
print("\n🔹 First 5 Rows:")
print(df.head())
print("\n🔹 Missing Values:")
print(df.isnull().sum())
print("\n🔹 Duplicates:", df.duplicated().sum())
```

```
0               Life Insurance             NaN
1  Product Specific Information             NaN
2            Installation/demo             NaN
3       Reverse Pickup Enquiry             NaN
4                   Not Needed             NaN

                             Order_id order_date_time Issue_reported at  \
0  c27c9bb4-fa36-4140-9f1f-21009254ffdb             NaN  01/08/2023 11:13
1  d406b0c7-ce17-4654-b9de-f08d421254bd             NaN  01/08/2023 12:52
2  c273368d-b961-44cb-beaf-62d6fd6c00d5             NaN  01/08/2023 20:16
3  5aed0059-55a4-4ec6-bb54-97942092020a             NaN  01/08/2023 20:56
4  e8bed5a9-6933-4aff-9dc6-ccefd7dcde59             NaN  01/08/2023 10:30

    issue_responded Survey_response_Date Customer_City Product_category  \
0  01/08/2023 11:47           01-Aug-23           NaN              NaN
1  01/08/2023 12:54           01-Aug-23           NaN              NaN
```

```
Duplicates: 0
```

```
df['Customer Remarks'].fillna("No Remark")
df['Customer_City'].fillna("Not Mentioned")
```

	Customer_City
0	Not Mentioned
1	Not Mentioned
2	Not Mentioned
3	Not Mentioned
4	Not Mentioned
...	...
85902	Not Mentioned
85903	Not Mentioned
85904	Not Mentioned
85905	Not Mentioned
85906	Not Mentioned

85907 rows × 1 columns

dtype: object

```
# Drop duplicate rows
df = df.drop_duplicates()
```

```
# 🧩 Step 3: Label Encode categorical columns
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
for col in ['Customer_City','Product_category']:
    df[col] = encoder.fit_transform(df[col])

# 🧩 Step 4: Compute correlation with CSAT score
corr_matrix = df[['Customer_City','Product_category', 'CSAT Score']].corr()

# 🧩 Step 5: Display correlation results
print("\nCorrelation with CSAT Score:")
print(corr_matrix['CSAT Score'].sort_values(ascending=False))
```

```
Correlation with CSAT Score:
CSAT Score          1.000000
Customer_City       0.083098
Product_category    0.076064
Name: CSAT Score, dtype: float64
```

```
df = df.drop(['Unique id','Order_id','order_date_time','Survey_response_Date','connected_handling_time','Item_price','Agent_na
```

```
# Fill numeric missing values with median
for col in df.select_dtypes(include=['int64', 'float64']).columns:
    df[col] = df[col].fillna(df[col].median())
```

```
df.isna().sum()
```

	0
channel_name	0
category	0
Sub-category	0
Customer Remarks	57165
Issue_reported at	0
issue_responded	0
Customer_City	0
Manager	0
Tenure Bucket	0
Agent Shift	0
CSAT Score	0

dtype: int64

```
df['Customer_City'].unique()
```

```
array(['NAGPUR', 'RANCHI', 'BETIA', 'NEW DELHI', 'KODERMA',
       'FARIDABAD', 'KANYAKUMARI', 'BURDWAN', 'JHUJHUNU', 'RANGAREDDY',
       'DHENKANAL', 'VERAVAL', 'PATTAMUNDAI', 'PALAMU', 'HARIDWAR',
       'GAYA', 'RASULABAD', 'VIRAMGAM', 'DHULE', 'BHUBANESWAR',
       'CHANGANACHERRY', 'BANGALORE', 'DARBHANGA', 'HALDWANI', 'PUNE',
       'CHANDRAPUR', 'JAMNAGAR', 'CUDDAPAH', 'AURANGABAD', 'MUNGER',
       'BARAUT', 'GOPALGANJ', 'MUZAFFARNAGAR', 'UNNAO', 'SAWER',
       'DHARAMASALA', 'AGRA', 'BERHAMPUR', 'UDAIPUR', 'BELLARY',
       'BANKURA', 'JALALPUR', 'JHUMRI TELAIYA', 'CUTTACK', 'JAJPUR ROAD',
       'HYDERABAD', 'JHANSI', 'BHIWANDI', 'DIMAPUR', 'BETTIAH',
       'SRINAGAR', 'VIDISHA', 'BILASPUR', 'LUCKNOW', 'PURNIA', 'KOLKATA',
       'MIRZAPUR', 'MADHUBANI', 'GHAZIABAD', 'AMBAD', 'KHAMGAON',
       'AMRAOTI', 'SRIPERUMBUDUR', 'BASTI', 'NALGONDA', 'BARMER',
       'CHARKHI DADRI', 'TIRUPATI', 'MUMBAI', 'BHOPAL', 'KATIHAR',
       'KHANDWA', 'MEDINIPUR', 'AHMEDABAD', 'CHENNAI', 'DURGAPUR',
       'BHIWADI', 'AKOLA', 'VISHAKHAPATNAM', 'ALLAHABAD', 'ULHASNAGAR',
       'AJMER', 'NAVI MUMBAI', 'JODHPUR', 'HATTA', 'ALWAR', 'ANAND',
       'AGARTALA', 'MATHURA', 'PALAKKAD', 'NAINITAL', 'SURAT', 'Dhansura',
       'CHATRAPUR', 'MORBI', 'AZAMGARH', 'HAMIRPUR', 'NAWADA',
       'BALESHWAR', 'VADODARA', 'CHIKHLI', 'GUNTUR', 'THANE',
       'VIJAYAWADA', 'BIDHUNA', 'AMBEDKAR NAGAR', 'GIRIDIH', 'GURGAON',
       'DOHAD', 'PURI', 'GANDHINAGAR', 'GAUTAM BUDDHA NAGAR', 'FAIZABAD',
       'GOLAGHAT', 'Garur', 'AMRELI', 'GUWAHATI', 'BALLIA',
       'GREATER NOIDA', 'KURNOOL', 'SAMASTIPUR', 'JAJPUR', 'JAMSHEDPUR',
       'DHOLPUR', 'PANVEL', 'CHIRAWA', 'COIMBATORE', 'BAREILLY',
       'MALIHABAD', 'KATHUA', 'BOKARO', 'NOKHA', 'PUTTUR', 'PURULIA',
       'KISHANGANJ', 'JADCHERLA', 'SHAJAPUR', 'ALIGARH', 'ARARIA',
       'TIJARA', 'RAIPUR', 'SANGLI', 'BIKANER', 'NARNAUND', 'BARDOLI',
       'KALOL', 'GONDAL', 'BHADOHI', 'HAZARIBAGH', 'PALAMANER', 'ROHTAK',
       'KOILWAR', 'AMBERNATH', 'INDORE', 'NAJIBABAD', 'GAIRKATA',
       'OSMANABAD', 'BEGUSARAI', 'KOTA', 'PATNA', 'JAIPUR', 'GORAKHPUR',
       'BIHAR SHARIF', 'VAPI', 'SOLAPUR', 'AMBIKAPUR', 'NANDED', 'ETAH',
       'BEED', 'RAJKOT', 'BHIWANI', 'ROURKELA', 'BHARTHANA', 'VARANASI',
       'NAGAUR', 'ALUVA', 'MADHEPURA', 'THRISSUR', 'CHAS', 'BHARUCH',
       'KANPUR', 'AIZAWL', 'VIRAR', 'DHANBAD', 'BHAGALPUR', 'ZAMANIA',
       'DARJILING', 'RAJAHMUNDRY', 'SILCHAR', 'RAJPUR', 'TUMKUR', 'CHURU',
       'JABALPUR', 'BIJNOR', 'GIDDERBAHA', 'MAHABUB NAGAR', 'SONIPAT',
       'NAWANSHAHAR', 'SIWAN', 'FATEHPUR', 'BULANDSHAHR', 'DEORIA',
       'MITHAPUR', 'SIKAR', 'ALIBAG', 'RAIPUR RANI', 'TINDIVANAM',
       'VEDASANDUR', 'MANDI', 'KOHIMA', 'BALUGAON', 'BANSWARA',
       'TULSIPUR', 'GANGAPUR CITY', 'RUDRA PRAYAG', 'JATNI', 'NELLORE',
       'JORHAT', 'TINSUKIA', 'VILLUPURAM', 'KULTI', 'SHAHAPUR', 'MAU',
       'MULBAGAL', 'MAHOBA', 'HAZIPUR', 'JHARSUGUDA', 'PINDWARA', 'PORSA',
       'JHARIA', 'KODUR', 'GOHANA', 'WANKANER', 'MUZAFFARPUR',
       'YAMUNANAGAR', 'DAUSA', 'MORADABAD', 'KANGRA', 'ERNAKULAM',
       'KISHTWAR', 'JANJGIR - CHAMPA', 'BELGAUM', 'WEST DINAJPUR',
       'SAKTI', 'BISWANATH CHARIALI', 'GANNAVARAM', 'KOTTAYAM',
       'BASAVAKALYAN', 'SAGAR', 'ASANSOL', 'SUNDERBANI', 'KURUKSHETRA',
       'NARWANA', 'LINGOTAM', 'TRICHY', 'PATTI', 'HASSAN', 'DAHOD',
       'PATHANAMTHITTA', 'UDAIPURWATI', 'RAE BARELI', 'ZIRAKPUR', 'DATIA',
       'SASARA', 'BUDNI', 'JUNAGADH', 'JHANJHARPUR', 'SARAN', 'Birbhum',
       'HUZURNAGAR', 'KOKRAJHAR', 'ALIPORE', 'BUXAR', 'SAHARANPUR',
       'BARIPARA', 'PHALODI', 'HINDAUN CITY', 'KANINA', 'DINDIGUL',
       'PILIBHIT', 'KAIRANA', 'KENDRAPARA', 'ANAPARTHI', 'BEHROR',
       'NALLASOPARA', 'SAHARSA', 'SATTENAPALLI', 'BERHAMPORE',
       'ANANDAPUR', 'ANANTAPUR', 'PINAHAT', 'UNA', 'BACHHRAWAN',
       'WARANGAL', 'MALDA', 'PANJIM', 'MAHESANA', 'AMARPUR', 'GWALIOR',
       'TIRUNELVELI', 'SAMBA', 'PANDUA', 'MADURAI', 'RECKONG PEO',
```

```
from textblob import TextBlob
# 🧩 Step 2: Define function for sentiment analysis
def get_sentiment(remark):
    # Keep "No remark" entries as they are
    if pd.isna(remark) or str(remark).strip().lower() == "no remark":
```

```
        return "No remark"

    # Use TextBlob to get polarity
    polarity = TextBlob(str(remark)).sentiment.polarity

    # Classify sentiment
    if polarity > 0.1:
        return "Positive"
    elif polarity < -0.1:
        return "Negative"
    else:
        return "Neutral"

# 🧩 Step 3: Apply sentiment function to the column
df['Sentiment'] = df['Customer Remarks'].apply(get_sentiment)
# 🧩 Step 5: Preview output
print(df[['Customer Remarks', 'Sentiment']].head())
```

```
  Customer Remarks  Sentiment
0              NaN  No remark
1              NaN  No remark
2              NaN  No remark
3              NaN  No remark
4              NaN  No remark
```

```
df = df.drop('Customer Remarks', axis=1)
```

```
df['Sentiment'].unique()
```

```
array(['No remark', 'Positive', 'Negative', 'Neutral'], dtype=object)
```

```
# Convert datetime columns to proper datetime format
df["Issue_reported at"] = pd.to_datetime(df["Issue_reported at"], format="%d/%m/%Y %H:%M")
df["issue_responded"] = pd.to_datetime(df["issue_responded"], format="%d/%m/%Y %H:%M")

# Calculate response time (in hours)
df["Response_Time_hours"] = (df["issue_responded"] - df["Issue_reported at"]).dt.total_seconds() / 3600

# Optionally, round off for neatness
df["Response_Time_hours"] = df["Response_Time_hours"].round(2)

# Display first few rows to verify
print(df[["Issue_reported at", "issue_responded", "Response_Time_hours"]].head())
df = df.drop(columns=['Issue_reported at', 'issue_responded'], errors='ignore')
# Save updated dataset
df.to_csv("CSAT_clean_data.csv", index=False)
print("✅ Response time added and saved successfully!")
```

```
    Issue_reported at     issue_responded  Response_Time_hours
0 2023-08-01 11:13:00 2023-08-01 11:47:00                 0.57
1 2023-08-01 12:52:00 2023-08-01 12:54:00                 0.03
2 2023-08-01 20:16:00 2023-08-01 20:38:00                 0.37
3 2023-08-01 20:56:00 2023-08-01 21:16:00                 0.33
4 2023-08-01 10:30:00 2023-08-01 10:32:00                 0.03
✅ Response time added and saved successfully!
```

```
# STEP 3: FEATURE ENGINEERING
if 'Feedback_Text' in df.columns:
    df['feedback_length'] = df['Feedback_Text'].astype(str).apply(len)
    df['word_count'] = df['Feedback_Text'].astype(str).apply(lambda x: len(x.split()))
    df['sentiment_score'] = df['Feedback_Text'].astype(str).apply(lambda x: TextBlob(x).sentiment.polarity)

# --- 3.3 Frequency Encoding for Categorical Columns ---
categorical_cols = df.select_dtypes(include=['object']).columns.tolist()
categorical_cols = [c for c in categorical_cols if c != 'CSAT']

for col in categorical_cols:
    freq_encoding = df[col].value_counts(normalize=True)
    df[col + '_freq'] = df[col].map(freq_encoding)
```