AI/ML Project Guide: Customer Sentiment Analysis for IBM AICTE Internship

Hello Ridhi! Based on the detailed summaries of the two technical sessions from your internship (Second Session on Jan 29, 2026, and Third Session on Feb 5, 2026), I see you're learning foundational ML concepts and hands-on implementation using Google Colab. The sessions emphasize project requirements: individual AI/ML projects in the AI domain, using techniques like supervised learning (classification for sentiment), data preprocessing, and tools like Colab and Kaggle datasets.

Your main project is on **AI/ML Customer Sentiment Analysis** (using text, voice, video, and generative AI for brands/companies), with the "Customer_Sentiment.csv" dataset. This fits perfectly as a supervised classification task (predicting sentiment: positive/negative/neutral), similar to the Employee Salary Prediction example in the third session (binary classification: income >50K or <=50K).

I'll help you build this project step-by-step, following the structure from the sessions:

1. **Project Setup & Guidelines** (inspired by Session 2).
2. **Data Collection & Loading** (using your CSV, like Kaggle Adult dataset).
3. **Data Preprocessing** (handling missing values, categories, outliers – detailed in Session 3).
4. **Feature Engineering** (encoding, scaling – like LabelEncoder in Session 3).
5. **Model Training & Evaluation** (supervised learning: classification models).
6. **Advanced AI Features** (add voice/video/generative AI as per your original plan).
7. **Deployment & Insights** (using Streamlit/FastAPI, as discussed earlier).

This will be a complete, end-to-end project in Colab. You can submit it as a PPT with screenshots/code explanations, as mentioned in the sessions.

1. Project Requirements & Alignment with Sessions

- **Domain**: AI/ML techniques (NLP for text sentiment, supervised classification).
- **Problem Statement**: Build an AI system to analyze customer sentiments from reviews (text), voice, and video inputs, generating insights for brands. Use your CSV for training/evaluation.
- **Why this fits**: Like the salary prediction, it's classification (sentiment labels). Incorporates ML types from Session 2 (supervised for main model, generative for insights).
- **Tools**: Google Colab (as taught), Pandas/Seaborn/Matplotlib (for exploration/visuals), Scikit-learn (for ML), Hugging Face (for advanced NLP), Librosa/SpeechRecognition (voice), OpenCV/FER (video).
- **Output**: Colab notebook, model accuracy >85%, visualizations, generative insights.
- **Innovations**: Multimodal (text+voice+video), generative AI for recommendations.

Upload your "Customer_Sentiment.csv" to Colab for this.

2. Google Colab Setup

Create a new Colab notebook (File > New Notebook). Install libraries (run in a cell):

```
Python

!pip install transformers torch torchaudio speechrecognition
librosa opencv-python-headless fer pandas matplotlib seaborn
scikit-learn gtts pydub
```

Import libraries (add to your notebook):

```
Python

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score,
classification_report
from transformers import pipeline
import speech_recognition as sr
import librosa
import cv2
from fer import FER
from gtts import gTTS
from pydub import AudioSegment
from google.colab import files
import io
import numpy as np
```

3. Data Collection & Loading

Like Session 3 (loading Adult CSV from Kaggle), upload your CSV:

```
Python

# Upload CSV
uploaded = files.upload()
df =
pd.read_csv(io.BytesIO(uploaded['Customer_Sentiment.csv']))
```

```
# Exploration (like .head(), .shape, .isnull().sum())
print("Data Shape:", df.shape)  # (25000, 13)
print(df.head())
print("\nMissing Values:\n", df.isnull().sum())  # Should be
none, as in Adult dataset example
```

Key features (like Adult's 15 columns):

- Numerical: customer_rating, response_time_hours, age_group (after encoding).
- Categorical: gender, region, product_category, platform, sentiment, review_text.
- Target: sentiment (positive/negative/neutral) or customer_rating (regression option).

4. Data Preprocessing (Hands-On like Session 3)

Follow the steps: cleaning categories, handling ?, outliers, redundancy.

```
Python

# A. Handle Missing Values (like Session 3: no nulls, but
check)
df = df.dropna()  # If any

# B. Categorical Cleaning (replace '?' or irrelevant, like
workclass in Adult)
# Example: Assume 'review_text' has no '?', but for
platform/workclass-like columns
df['platform'] = df['platform'].replace('?', 'Others')  # If
any
# Remove irrelevant categories (e.g., if any low-count
platforms)
platform_counts = df['platform'].value_counts()
low_count_platforms = platform_counts[platform_counts <
10].index
df = df[~df['platform'].isin(low_count_platforms)]  # Like
removing 'Never-worked'

# C. Outlier Detection (Box Plot like Session 3 for age)
plt.figure(figsize=(10, 6))
```

```
sns.boxplot(x=df['response_time_hours'])  # Example for
response time
plt.title('Outliers in Response Time')
plt.show()

# Remove outliers (e.g., response_time > 100 hours unlikely)
Q1 = df['response_time_hours'].quantile(0.25)
Q3 = df['response_time_hours'].quantile(0.75)
IQR = Q3 - Q1
df = df[(df['response_time_hours'] >= (Q1 - 1.5 * IQR)) &
(df['response_time_hours'] <= (Q3 + 1.5 * IQR))]

# D. Handle Redundancy (drop if any, like education/education-
num)
# No direct redundancy, but drop 'customer_id' as irrelevant
df = df.drop(['customer_id'], axis=1)

print("Cleaned Data Shape:", df.shape)
```

5. Feature Engineering (Encoding & Scaling like Session 3)

Use LabelEncoder for categories (as taught: simpler than One-Hot for this).

Python

```
# A. Label Encoding (for all categorical columns)
categorical_cols = ['gender', 'age_group', 'region',
'product_category', 'purchase_channel', 'platform',
'sentiment', 'issue_resolved', 'complaint_registered']
le_dict = {}
for col in categorical_cols:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    le_dict[col] = le  # Save for inverse transform later

# Target: sentiment (already encoded: 0=negative, 1=neutral,
2=positive)

# B. Data Splitting (X=features, Y=target like Session 3)
X = df.drop(['sentiment', 'review_text'], axis=1)  # Exclude
text for basic ML; use later for NLP
y = df['sentiment']
```

```
# C. Scaling (MinMaxScaler like Session 3)
scaler = MinMaxScaler()
X_scaled = pd.DataFrame(scaler.fit_transform(X),
columns=X.columns)

# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X_scaled,
y, test_size=0.2, random_state=42)
```

6. Model Training & Evaluation (Supervised Classification)

Use Logistic Regression (as mentioned in Session 2 for classification).

Python

```
# Train Model
model = LogisticRegression(multi_class='multinomial',
max_iter=200)
model.fit(X_train, y_train)

# Predict & Evaluate
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
print(classification_report(y_test, y_pred))

# Visualization: Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix')
```

```
plt.show()
```

For advanced NLP on 'review_text' (AI domain focus):

Use Hugging Face pipeline (fine-tuned as in previous code).

```
Python

sentiment_pipeline = pipeline("sentiment-analysis")
df['predicted_sentiment_text'] = df['review_text'].apply(lambda
x: sentiment_pipeline(x)[0]['label'])
# Compare with original sentiment
```

7. Advanced Features: Voice/Video/Generative AI

Add demos as per your plan (generate audio if needed, as I helped earlier).

- **Voice Demo** (upload or generate):

```
Python

# Generate sample audio (from earlier help)
review_text = "This product is amazing!"
tts = gTTS(review_text)
tts.save("sample.mp3")
audio = AudioSegment.from_mp3("sample.mp3")
audio.export("sample.wav", format="wav")

# Analyze function (as before)
def analyze_voice_sentiment(audio_data):
    # ... (full code from previous)
# Upload and run
```

- **Video Demo**: Similar upload and FER analysis.

- Make Better: Similar upload and PDF analysis.

- **Generative AI Insights**:

```
Python

from transformers import GPT2LMHeadModel, GPT2Tokenizer
tokenizer = GPT2Tokenizer.from_pretrained("gpt2")
gen_model = GPT2LMHeadModel.from_pretrained("gpt2")

def generate_insights(sentiment):
    prompt = f"Based on sentiment {sentiment}, suggest
brand improvements:"
    inputs = tokenizer.encode(prompt, return_tensors="pt")
    outputs = gen_model.generate(inputs, max_length=100)
    return tokenizer.decode(outputs[0])

print(generate_insights("negative"))
```

8. Deployment & Final Tips

- **Streamlit App**: As previous, wrap in app.py and deploy via GitHub.
- **PPT Submission**: Include screenshots of Colab cells, accuracy, graphs. Explain ML types used (supervised classification).
- **Next Steps**: Fine-tune on full data, add unsupervised clustering for customer segments (from Session 2).
- **Resources**: Use Kaggle for similar datasets, Colab for all work (no VS Code needed yet).

Run this in Colab – it should take 10-20 mins. If stuck (e.g., errors), share the output. Good luck with your internship in Meerut! If you need Hindi explanations or local examples, let me know.