TF-IDF PRACTICAL

```
from sklearn.feature_extraction.text import TfidfVectorizer

sentences = [
    "I love AI",
    "AI is very powerful",
    "I love machine learning"
]

vectorizer = TfidfVectorizer()

X = vectorizer.fit_transform(sentences)

print("Words in Vocabulary:", vectorizer.get_feature_names_out())
print("\nTF-IDF Matrix:\n")
print(X.toarray())
```

- **TfidfVectorizer()**
  - **Yeh object TF-IDF generate karta hai.**
- **fit_transform(sentences)**
  - **fit = vocabulary banana**
  - **transform = text ko numbers me convert karna**

- **get_feature_names_out()**
  - **TF-IDF me jo words use hue, unka list deta hai**
- **X.toarray()**
  - **sparse matrix ko proper array me convert karta hai**

Sentiment Analysis

PROJECT FLOW

- **Dataset**
- ↓
- **Text Cleaning**
- ↓
- **TF-IDF Conversion**
- ↓
- **Train-Test Split**
- ↓
- **ML Model Training**
- ↓
- **Accuracy Check**
- ↓
- **User Input Prediction**
- **Dataset:**

Text	Label
I love this product	1
This is amazing	1
I am very happy	1
I hate this	0
This is very bad	0
I am disappointed	0

📌 1 = Positive
📌 0 = Negative

```
import pandas as pd
import nltk
import string

from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB

# Download required nltk data
nltk.download('punkt')
nltk.download('stopwords')

# ------------------------------
# Step 1: Dataset
# ------------------------------
```

```
data = {
    "text": [
        "I love this product",
        "This is amazing",
        "I am very happy",
        "I hate this",
        "This is very bad",
        "I am disappointed"
    ],
    "label": [1, 1, 1, 0, 0, 0]
}

df = pd.DataFrame(data)

print("Original Dataset:\n")
print(df)

# ------------------------------
# Step 2: Text Cleaning Function
# ------------------------------
def clean_text(text):
    text = text.lower()
```

```
    tokens = word_tokenize(text)
    tokens = [word for word in tokens if word not in
string.punctuation]

    stop_words = stopwords.words('english')
    tokens = [word for word in tokens if word not in
stop_words]

    return " ".join(tokens)

# Apply cleaning
df["clean_text"] = df["text"].apply(clean_text)

print("\nCleaned Dataset:\n")
print(df)

# ----------------------------
# Step 3: TF-IDF Vectorization
# ----------------------------
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(df["clean_text"])
y = df["label"]
```

```
print("\nTF-IDF Words (Vocabulary):\n")
print(vectorizer.get_feature_names_out())

print("\nTF-IDF Matrix:\n")
print(X.toarray())

# ------------------------------
# Step 4: Train Model on FULL Dataset
# ------------------------------
model = MultinomialNB()
model.fit(X, y)

# ------------------------------
# Step 5: User Input Prediction
# ------------------------------
user_sentence = input("\nEnter a sentence for sentiment prediction: ")

clean_user = clean_text(user_sentence)
user_vector = vectorizer.transform([clean_user])
```

```
prediction = model.predict(user_vector)


print("\nYour sentence:", user_sentence)


if prediction[0] == 1:
    print("Sentiment: Positive ☺")
else:
    print("Sentiment: Negative ☹")
```