CLASSIFYING REDDIT POSTS USING RANDOM FOREST

Python Code:

```
import os
import re
import pandas as pd
from sklearn.feature_extraction.text import ENGLISH_STOP_WORDS, TfidfVectorizer
from sklearn.metrics import accuracy_score, classification_report
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier



# ------------------------------------------------------------
# 1. CONSTANTS & FILE PATH
# ------------------------------------------------------------
DATA_PATH = "/Users/rajeshmanikandans/Desktop/M.Tech Class/DSE/bike_reviews_dataset.csv"
MAX_FEATURES = 5000    # TF-IDF vocabulary size

# Fallback data if dataset is missing
SAMPLE_FALLBACK = [
    {"text": "This bike is fast and lightweight.", "label": "Sports Bike"},
    {"text": "Battery backup is great for long commutes.", "label": "Electric Bike"},
]

STOP_WORDS = set(ENGLISH_STOP_WORDS)



# ------------------------------------------------------------
# 2. LOAD DATASET
# ------------------------------------------------------------
if os.path.exists(DATA_PATH):
    df = pd.read_csv(DATA_PATH)
    print("Dataset loaded successfully!\n")
else:
    print("Dataset not found. Using fallback sample dataset.\n")
    df = pd.DataFrame(SAMPLE_FALLBACK)

# Check required columns
required_columns = {"text", "label"}
```

```
missing_cols = required_columns.difference(df.columns)
if missing_cols:
    raise ValueError(f"ERROR: Dataset missing required columns: {missing_cols}")

print("Sample Rows:\n", df.head(), "\n")



# ----------------------------------------------------------
# 3. TEXT CLEANING FUNCTION
# ----------------------------------------------------------
def clean_text(text: str) -> str:
    text = text.lower()
    text = re.sub(r"http\\S+", " ", text)          # Remove URLs
    text = re.sub(r"[^a-z\\s]", " ", text)         # Remove digits & punctuation
    text = re.sub(r"\\s+", " ", text).strip()      # Remove extra spaces

    tokens = [w for w in text.split() if w not in STOP_WORDS]  # Remove stopwords
    return " ".join(tokens)

# Apply cleaning
df["clean_text"] = df["text"].apply(clean_text)
print("Clean Text:\n", df["clean_text"].head(), "\n")



# ----------------------------------------------------------
# 4. TRAIN–TEST SPLIT
# ----------------------------------------------------------
X_train, X_test, y_train, y_test = train_test_split(
    df["clean_text"], df["label"],
    test_size=0.2,
    random_state=42
)

print(f"Training Samples: {len(X_train)}")
print(f"Testing Samples: {len(X_test)}\n")



# ----------------------------------------------------------
# 5. TF-IDF VECTORIZATION
# ----------------------------------------------------------
tfidf = TfidfVectorizer(max_features=MAX_FEATURES)
X_train_vec = tfidf.fit_transform(X_train)
X_test_vec = tfidf.transform(X_test)   # Use the same TF-IDF model
```

```
# ----------------------------------------------------------
# 6. TRAIN RANDOM FOREST MODEL
# ----------------------------------------------------------
model = RandomForestClassifier(
    n_estimators=200,
    random_state=42,
    n_jobs=-1
)

model.fit(X_train_vec, y_train)



# ----------------------------------------------------------
# 7. MODEL EVALUATION
# ----------------------------------------------------------
y_pred = model.predict(X_test_vec)

print("\nModel Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n")
print(classification_report(y_test, y_pred, zero_division=0))



# ----------------------------------------------------------
# 8. CLASSIFY USER-PROVIDED REDDIT POST
# ----------------------------------------------------------
user_input = input("\nEnter a Reddit post to classify: ")

# Clean & transform user input
clean_user = clean_text(user_input)
vector_user = tfidf.transform([clean_user])

# Predict category
predicted_label = model.predict(vector_user)[0]

print("\nReddit Post:", user_input)
print("Predicted Category:", predicted_label)
```

Output :

```
Sample Rows:
                                      text          label
0          This bike is fast and lightweight.    Sports Bike
1  Battery backup is great for long commutes.  Electric Bike

Clean Text:
 0                bike fast lightweight
1    battery backup great long commutes
Name: clean_text, dtype: object

Training Samples: 1
Testing Samples: 1


Model Accuracy: 0.0

Classification Report:

               precision    recall  f1-score   support

Electric Bike       0.00      0.00      0.00       1.0
  Sports Bike       0.00      0.00      0.00       0.0

     accuracy                           0.00       1.0
    macro avg       0.00      0.00      0.00       1.0
 weighted avg       0.00      0.00      0.00       1.0


Enter a Reddit post to classify: This bike is amazing for long rides and smooth performance!

Reddit Post: This bike is amazing for long rides and smooth performance!
Predicted Category: Sports Bike
```

EXPLANATION

1. Objective

The objective of this assignment is to build a **text classification model** that categorizes **Reddit-style bike reviews** into different types (Sports, Electric, Adventure, etc.).
We use:
✔ **TF-IDF vectorization**
✔ **Random Forest Classifier**

2. Dataset Description

The dataset contains the following columns:

Column	Description
text	The bike review written by the user
label	Category (Sports Bike, Electric Bike, etc.)

A fallback dataset is used if the CSV is missing.

3. Data Preprocessing

a. Lowercasing

Converts text to lowercase.

b. Removing URLs

Examples like:
https://bike.com/model → removed

c. Removing Punctuation & Numbers

Text becomes alphabet-only.

d. Removing Stopwords

Words like "the", "is", "and" are removed.

e. Final Output Example

Input:

The mileage is very good!!

Output:

mileage good

4. Train-Test Split

The dataset is split:

Set	Percentage
Train	80%
Test	20%

5. TF-IDF Vectorization

TF-IDF converts text into numerical vectors using:

- Term Frequency
- Inverse Document Frequency
- Maximum 5000 features

Output is a matrix of vectorized text.

6. Random Forest Classification

We use:

RandomForestClassifier(n_estimators=200)

Reasons:

- High performance
- Works well with TF-IDF
- Handles noisy text
- Easy to train and interpret

7. Evaluation

Metrics used:

- **Accuracy**
- **Precision**
- **Recall**
- **F1-score**

A classification report is printed.

8. Predicting a Reddit Post

The user types any bike-related Reddit post.

Example:

This bike is amazing for long rides and smooth performance!

Output:

Predicted Category: Sports Bike

⭐ BLOCK DIAGRAM