The code builds a sentiment analysis web application using Streamlit and a pre-trained model from the `transformers` library.

Imports

```
import streamlit as st
```

- Imports the `streamlit` library, aliased as `st`, which is used to create interactive web applications in Python.

```
from transformers import pipeline
```

- Imports the `pipeline` function from the `transformers` library (by Hugging Face), which provides an easy way to use pre-trained machine learning models, such as for sentiment analysis.

Model Loading Function

```
@st.cache_resource
def load_model():
    return pipeline("sentiment-analysis", model="cardiffnlp/twitter-roberta-base-sentiment")
```

- `@st.cache_resource`: A Streamlit decorator that caches the result of the `load_model` function to avoid reloading the model on every interaction, improving performance.
- `def load_model()`: Defines a function that loads the sentiment analysis model.
- `return pipeline(...)`: Returns a pre-trained sentiment analysis pipeline using the `cardiffnlp/twitter-roberta-base-sentiment` model, specifically trained for Twitter sentiment analysis with three classes (positive, neutral, negative).

Constants

```
COLOR_MAP = {
    "POSITIVE": "#90EE90",
    "NEGATIVE": "#FF6B6B",
    "NEUTRAL": "#FFD93D"
}
```

- Defines a dictionary `COLOR_MAP` that maps sentiment labels to HEX color codes:
    - `POSITIVE`: Light green (#90EE90)
    - `NEGATIVE`: Light red (#FF6B6B)
    - `NEUTRAL`: Light yellow (#FFD93D)

```
EMOJI_MAP = {
    "POSITIVE": "😊",
    "NEGATIVE": "😠",
    "NEUTRAL": "😐"
}
```

- Defines a dictionary `EMOJI_MAP` that maps sentiment labels to corresponding emojis:
  - `POSITIVE`: Smiling face (😊)
  - `NEGATIVE`: Angry face (😠)
  - `NEUTRAL`: Neutral face (😐)

Page Configuration

```
st.set_page_config(page_title="Sentiment Analysis", layout="wide")
```

- Configures the Streamlit app:
  - `page_title`: Sets the browser tab title to "Sentiment Analysis".
  - `layout="wide"`: Uses a wide layout for the app interface.

```
st.title("▯ Dynamic Colorful Sentiment Analysis")
```

- Displays a title at the top of the app with an art palette emoji (▯) and the text "Dynamic Colorful Sentiment Analysis".

Sidebar

```
with st.sidebar:
    st.header("⚙ Settings")
    st.markdown("""
    **Used Model:**
    cardiffnlp/twitter-roberta-base-sentiment
    (3-class Twitter sentiment analysis model)

    **Color Coding:**
    - ▯ Positive: #90EE90 (Light Green)
    - ▯ Negative: #FF6B6B (Light Red)
    - ▯ Neutral: #FFD93D (Light Yellow)
    """)
```

- `with st.sidebar:`: Creates a sidebar section in the app.
- `st.header("⚙ Settings")`: Adds a header with a gear emoji (⚙) and the text "Settings".
- `st.markdown(...)`: Displays formatted text in the sidebar using Markdown:
  - Specifies the model used (`cardiffnlp/twitter-roberta-base-sentiment`).
  - Lists the color coding for each sentiment with emojis and HEX codes.

```
    st.header("ℹ Information Panel")
    st.markdown("""
    **Application Features:**
    - Real-time sentiment analysis based on text input
    - Emotion-specific dynamic background color
    - Sentiment label and confidence score display
    - Color transition animation
    """)
```

- Adds another sidebar section with a header "ℹ Information Panel" (info emoji).
- Lists the app's features in Markdown format, such as real-time analysis and dynamic color changes.

Main Input and Logic

```
user_input = st.text_input("Enter text:", "")
```

- Creates a text input box labeled "Enter text:" where the user can type. The input is stored in the `user_input` variable, initialized as an empty string.

```
if user_input:
```

- Checks if the user has entered any text. If true, the code inside this block runs.

```
    classifier = load_model()
```

- Calls the `load_model()` function to load the sentiment analysis model and assigns it to `classifier`.

```
    result = classifier(user_input)[0]
```

- Runs the sentiment analysis on `user_input` using the `classifier`. The result is a list of dictionaries, and `[0]` extracts the first (and only) result, which contains `label` and `score`.

```
    label_num = int(result['label'].split("_")[-1])
    label = ["NEGATIVE", "NEUTRAL", "POSITIVE"][label_num]
```

- `result['label']`: The model returns labels like `LABEL_0`, `LABEL_1`, or `LABEL_2`.
- `split("_")[-1]`: Splits the label string at "_" and takes the last part (e.g., "0", "1", or "2").
- `int(...)`: Converts the number to an integer.
- Maps the number to a human-readable label:
  - 0 → "NEGATIVE"
  - 1 → "NEUTRAL"
  - 2 → "POSITIVE"

```
    st.markdown(
        f"""
        <style>
            [data-testid="stAppViewContainer"] > .main {{
                background-color: {COLOR_MAP[label]};
                transition: background-color 0.5s ease;
            }}
            [data-testid="stHeader"] {{
                background-color: rgba(0,0,0,0);
            }}
        </style>
        """,
        unsafe_allow_html=True
    )
```

- Injects custom CSS into the app using `st.markdown` with `unsafe_allow_html=True`:
  - Changes the background color of the main app container to the color from `COLOR_MAP` based on the sentiment label.
  - Adds a 0.5-second smooth color transition effect (`transition`).
  - Makes the header background transparent (`rgba(0,0,0,0)`).

```
    col1, col2 = st.columns(2)
```

- Creates two columns in the app layout to display the result and confidence score side by side.

```
    with col1:
        st.subheader(f"{EMOJI_MAP[label]} Result: {label}")
```

- In the first column, displays a subheader with the emoji from `EMOJI_MAP` and the sentiment label (e.g., "😊 Result: POSITIVE").

```
    with col2:
        st.metric("Confidence Score", f"{result['score']:.2%}")
```

- In the second column, displays the model's confidence score as a percentage (e.g., "Confidence Score: 92.34%").
- `result['score']`: The raw confidence score (0 to 1).
- `.2%`: Formats it as a percentage with 2 decimal places.

```
else:
    st.info("Please enter some text")
```

- If `user_input` is empty, displays an info message prompting the user to enter text.

Help Section Function

```python
def show_help_section():
    st.sidebar.subheader("□ Help & Information")
```

- Defines a function `show_help_section()` to display help content in the sidebar.
- Adds a subheader with a book emoji (□) and the text "Help & Information".

```python
    st.sidebar.markdown("""
    **Confidence Score:**
    Model's confidence in prediction (0-1 range, higher = more
confident)

    **Quick Guide:**
    - 😊 Positive Sentiment
    - 😠 Negative Sentiment
    - 😐 Neutral Sentiment
    - Colors change automatically based on sentiment
    """)
```

- Adds basic help information in the sidebar:
  - Explains the confidence score range (0 to 1).
  - Provides a quick guide with emojis and a note about automatic color changes.

```python
    with st.sidebar.expander("□ Detailed Technical Information"):
        st.markdown("""
        **Confidence Score**
        - **Calculation Method:** Directly taken from model outputs
(result['score'])
        - **Interpretation:**
          0.0-0.4 → Low confidence
          0.4-0.6 → Medium confidence
          0.6-1.0 → High confidence

        **Color Codes**
        | Sentiment | HEX Code     | Example      |
        |-----------|-------------|-------------|
        | Positive  | #90EE90    | □ Light Green |
        | Negative  | #FF6B6B    | □ Light Red |
        | Neutral   | #FFD93D    | □ Light Yellow |

        **Emoji Symbolism**
        - 😊 → Positive words/expressions
        - 😠 → Derogatory or angry expressions
        - 😐 → Emotionally neutral content
        """)
```

- Creates an expandable section in the sidebar labeled "□ Detailed Technical Information" (chart emoji).
- Provides detailed info:

  - How the confidence score is calculated and its interpretation ranges.
  - A table of sentiment color codes with HEX values and examples.
  - Explanation of what each emoji represents.

Adding Help Section to Sidebar

```python
with st.sidebar:
    show_help_section()
```

- Calls the `show_help_section()` function to display the help content in the sidebar.

Summary

This code creates a Streamlit web app that:

1. Takes user text input.
2. Analyzes its sentiment using a pre-trained Twitter sentiment model.
3. Displays the result with an emoji, label, and confidence score.
4. Changes the background color dynamically based on the sentiment.
5. Provides a sidebar with settings and detailed help information.