```
import streamlit as st
import pandas as pd
import pickle
from sklearn.preprocessing import LabelEncoder, StandardScaler
from pathlib import Path
import os

# Get the current directory
current_dir = Path(__file__).resolve().parent

# # List the contents of the directory
# directory_contents = os.listdir(current_dir)

# # Display the contents using st.write
# st.write(f"Contents of Directory '{current_dir}':")
# st.write(directory_contents)

# Navigate back to the main project directory
project_dir = current_dir.parent

# Streamlit app
# set page title
st.set_page_config('Car Price Prediction App')

# Hide Streamlit's GitHub icon

# Hide Streamlit's GitHub icon
hide_st_style = """
            <style>
            #MainMenu {visibility: hidden;}
            header {visibility: hidden;}
            footer {visibility: hidden;}
            </style>
            """
st.markdown(hide_st_style, unsafe_allow_html=True)

# Side Nav-bar window
social_acc = ['About', 'Github', 'Kaggle', 'LinkedIn']
social_acc_nav = st.sidebar.selectbox('About', social_acc)
if social_acc_nav == 'About':
    st.sidebar.markdown("<h2 style='text-align: center;'> Sumit S. Chaure</h2> ",
unsafe_allow_html=True)
    st.sidebar.markdown('''---''')
    st.sidebar.markdown('''
    • Data Analytics (Python/SQL/Power Bi/Excel)\n
    • Data Science (Python,Machine Learning,Statistics)\n
    • Interned as a Data Analyst @ Trainity\n
    • Working Experience of 2.5 Years as a Software Engineer''')
    st.sidebar.markdown("[Contact Me](mailto:sumitsc.work@gmail.com)")
elif social_acc_nav == 'Kaggle':
    st.sidebar.image('https://mitsus.life-is-pa.in/6ohYGkH40.png')
    st.sidebar.markdown("[Kaggle](https://www.kaggle.com/mitsu00)")
elif social_acc_nav == 'Github':
    st.sidebar.image('https://mitsus.life-is-pa.in/6ohZdGxSM.png')
    st.sidebar.markdown("[Github Profile](https://github.com/Sumit-SC)")
elif social_acc_nav == 'LinkedIn':
    st.sidebar.image('https://mitsus.life-is-pa.in/6ohYtyxiP.png')
    st.sidebar.markdown("[Visit LinkedIn
account](https://www.linkedin.com/in/sumitsc/)")
```

```
# Load the saved model
model_filename = current_dir / "./models/Random Forest.pkl"
loaded_model = pickle.load(open(model_filename, "rb"))

# import raw data & show
raw_df = pd.read_csv(current_dir / './dataset/raw/CAR DETAILS.csv')

# Load the cleaned data
cleaned_data_filename = current_dir / "./dataset/processed/Processed CAR
DETAILS.csv"
cleaned_data = pd.read_csv(cleaned_data_filename)

category_col = ['Brand', 'Model', 'Variant', 'Fuel', 'Seller_Type', 'Transmission',
'Owner']

# Main Page Starting

# Menu Bar/Page selection
menu_list = ['Raw-Data Display', 'Exploratory Data Analysis', "Predict Selling
Price"]
menu = st.radio("Menu", menu_list)

# Raw-Data Display
if menu == 'Raw-Data Display':
    st.title('Used Car Models Raw-Dataset')
    if st.checkbox("View Raw Car dataset"):
        st.write(raw_df)

# EDA Part(Will Add Later)
if menu == 'Exploratory Data Analysis':
    st.title('Exploratory Data Analysis of Used Car Models ')
    st.write("EDA Code to view & Explore the Graphs & inisghts :")
    st.write("[EDA](https://github.com/Sumit-SC/Data-Science-Capstone-Project/
blob/main/notebooks/Used_Cars_DA(Graphical%26Cleaning).ipynb) ,")

st.write("[Colab](https://colab.research.google.com/drive/1ObKA8DxCUC5S_2riq31XJoZr
fxLq8bbn#scrollTo=yt1XakQBL8tD)")
    if st.checkbox("View Cleaned/Processed data"):
        st.write(cleaned_data)

# Price Prediction
elif menu == 'Predict Selling Price':
    # Display the columns in the web app
    st.title("Car Selling Price Prediction App")

    # Display a dropdown to toggle between loaded CSV data and encoded data
    display_option = st.radio("Select Display Option:", ["No Data", "Loaded CSV
Data", "Encoded Data"])

    # function for encoding loaded dataset
    def preprocess_data(df, label_encoders):
        for feature in df.columns:
            if feature in label_encoders:
                df[feature] = label_encoders[feature].transform(df[feature])
        return df

    # Load the LabelEncoders used during training
    label_encoders = {}
```

```
    for feature in category_col:
        label_encoder = LabelEncoder()
        label_encoder.fit(cleaned_data[feature])
        label_encoders[feature] = label_encoder

    # Display loaded CSV data
    # st.subheader("Loaded CSV Data:")
    # st.write(cleaned_data)

    # Display encoded data
    # st.subheader("Encoded Data:")
    encoded_data = preprocess_data(cleaned_data.copy(), label_encoders)
    # st.write(encoded_data)

    # Display the selected data
    if display_option == "No Data":
        st.subheader("Not displaying either the Loaded CSV File nor the Encoded
Data")
    elif display_option == "Loaded CSV Data":
        st.subheader("Loaded CSV Data:")
        st.write(cleaned_data)
    elif display_option == "Encoded Data":
        st.subheader("Encoded Data:")
        st.write(encoded_data)

    # Display sliders for numerical features
    km_driven = st.slider("Select KM Driven:",
min_value=cleaned_data["Km_Driven"].min(),
                          max_value=cleaned_data["Km_Driven"].max())
    year = st.slider("Select Year:", min_value=cleaned_data["Year"].min(),
max_value=cleaned_data["Year"].max())

    # Display dropdowns for categorical features
    selected_brand = st.selectbox("Select Brand:", cleaned_data["Brand"].unique())
    brand_filtered_df = cleaned_data[cleaned_data['Brand'] == selected_brand]
    selected_model = st.selectbox("Select Model:",
brand_filtered_df["Model"].unique())
    model_filtered_df = brand_filtered_df[brand_filtered_df['Model'] ==
selected_model]
    selected_variant = st.selectbox("Select Variant:",
model_filtered_df["Variant"].unique())
    selected_fuel = st.selectbox("Select Fuel:", cleaned_data["Fuel"].unique())
    selected_seller_type = st.selectbox("Select Seller Type:",
cleaned_data["Seller_Type"].unique())
    selected_transmission = st.selectbox("Select Transmission:",
cleaned_data["Transmission"].unique())
    selected_owner = st.selectbox("Select Owner:", cleaned_data["Owner"].unique())

    # Create a DataFrame from the user inputs
    input_data = pd.DataFrame({
        'Brand': [selected_brand],
        'Model': [selected_model],
        'Variant': [selected_variant],
        'Year': [year],
        'Km_Driven': [km_driven],
        'Fuel': [selected_fuel],
        'Seller_Type': [selected_seller_type],
        'Transmission': [selected_transmission],
        'Owner': [selected_owner]
```

```
    })

    # Check if the loaded model and input data are correct
    st.subheader("Loaded Model:")
    st.write(loaded_model)

    st.subheader("Processed Input Data:")
    st.write(input_data)

    # Preprocess the user input data using the same label encoders
    input_data_encoded = preprocess_data(input_data.copy(), label_encoders)

    st.subheader("Processed Input Data:(After Encoding)")
    st.write(input_data_encoded)

    # Standardize numerical features using scikit-learn's StandardScaler
    scaler = StandardScaler()
    numerical_cols = ['Year', 'Km_Driven']
    input_data_encoded[numerical_cols] =
scaler.fit_transform(input_data_encoded[numerical_cols])

    # Make prediction using the loaded model
    if st.button("Predict Selling Price"):
        # Make predictions
        predicted_price = loaded_model.predict(input_data_encoded)
        st.subheader("Predicted Selling Price:")
        st.write(f"The predicted selling price is:
**_{predicted_price[0]:,.2f}_**")
```