import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeRegressor

from sklearn.preprocessing import StandardScaler, OneHotEncoder

from sklearn.metrics import mean\_squared\_error

from sklearn.model\_selection import GridSearchCV

from sklearn.preprocessing import LabelBinarizer

import streamlit as st

import re

#st.set\_page\_config(layout="wide")

st.set\_page\_config(layout="wide", initial\_sidebar\_state="collapsed")

import os

import pickle

#application background

def app\_bg():

    st.markdown(f""" <style>.stApp {{

                        background: url("https://tse2.mm.bing.net/th?id=OIP.-Omurt7RrwN6iJ2FfrSp3gHaFj&pid=Api&P=0&h=220");

                        background-size: cover}}

                     </style>""",unsafe\_allow\_html=True)

app\_bg()

st.write("""

<div style='text-align:center; background-color:#009999;'>

    <h1 style='color:#FFFFFF; font-family: Rockwell, sans-serif; font-weight: bold;'>Industrial Copper Modeling Application</h1>

</div>

""", unsafe\_allow\_html=True)

# Define custom CSS styles for the tabs

tab\_style = """

    font-size: 30px;

    font-family: Rockwell, Rockwell;

    color: black;

    text-align: center;

"""

# Create tabs

tab1, tab2 = st.columns(2)

# Render the tabs with custom styling

with tab1:

    st.markdown("<div style='" + tab\_style + "'>PREDICT SELLING PRICE</div>", unsafe\_allow\_html=True)

with tab2:

    st.markdown("<div style='" + tab\_style + "'>PREDICT STATUS</div>", unsafe\_allow\_html=True)

tab1, tab2 = st.tabs(["PREDICT SELLING PRICE", "PREDICT STATUS"])

with tab1:

        # Define the possible values for the dropdown menus

        status\_options = ['Won', 'Draft', 'To be approved', 'Lost', 'Not lost for AM', 'Wonderful', 'Revised', 'Offered', 'Offerable']

        item\_type\_options = ['W', 'WI', 'S', 'Others', 'PL', 'IPL', 'SLAWR']

        country\_options = [28., 25., 30., 32., 38., 78., 27., 77., 113., 79., 26., 39., 40., 84., 80., 107., 89.]

        application\_options = [10., 41., 28., 59., 15., 4., 38., 56., 42., 26., 27., 19., 20., 66., 29., 22., 40., 25., 67., 79., 3., 99., 2., 5., 39., 69., 70., 65., 58., 68.]

        product=['611112', '611728', '628112', '628117', '628377', '640400', '640405', '640665',

                     '611993', '929423819', '1282007633', '1332077137', '164141591', '164336407',

                     '164337175', '1665572032', '1665572374', '1665584320', '1665584642', '1665584662',

                     '1668701376', '1668701698', '1668701718', '1668701725', '1670798778', '1671863738',

                     '1671876026', '1690738206', '1690738219', '1693867550', '1693867563', '1721130331', '1722207579']

        # Define the widgets for user input

        with st.form("my\_form"):

            col1,col2,col3=st.columns([5,2,5])

            with col1:

                st.write(' ')

                status = st.selectbox("Status", status\_options,key=1)

                item\_type = st.selectbox("Item Type", item\_type\_options,key=2)

                country = st.selectbox("Country", sorted(country\_options),key=3)

                application = st.selectbox("Application", sorted(application\_options),key=4)

                product\_ref = st.selectbox("Product Reference", product,key=5)

            with col3:

                st.write( f'<h5 style="color:rgb(0, 153, 153,0.4);">NOTE: Min & Max given for reference, you can enter any value</h5>', unsafe\_allow\_html=True )

                quantity\_tons = st.text\_input("Enter Quantity Tons (Min:611728 & Max:1722207579)")

                thickness = st.text\_input("Enter thickness (Min:0.18 & Max:400)")

                width = st.text\_input("Enter width (Min:1, Max:2990)")

                customer = st.text\_input("customer ID (Min:12458, Max:30408185)")

                submit\_button = st.form\_submit\_button(label="PREDICT SELLING PRICE")

                st.markdown("""

                    <style>

                    div.stButton > button:first-child {

                        background-color: #009999;

                        color: white;

                        width: 100%;

                    }

                    </style>

                """, unsafe\_allow\_html=True)

            flag=0

            pattern = "^(?:\d+|\d\*\.\d+)$"

            for i in [quantity\_tons,thickness,width,customer]:

                if re.match(pattern, i):

                    pass

                else:

                    flag=1

                    break

        if submit\_button and flag==1:

            if len(i)==0:

                st.write("please enter a valid number space not allowed")

            else:

                st.write("You have entered an invalid value: ",i)

        if submit\_button and flag==0:

            import pickle

            model\_file\_path = "D:\\copper project\\model.pkl"

            with open(model\_file\_path, 'rb') as file:

                # Your code to handle the file

            #with open(r"model/model.pkl", 'rb') as file:

                loaded\_model = pickle.load(file)

            model\_file\_path = "D:\\copper project\\scaler.pkl"

            with open(model\_file\_path, 'rb') as file:

            #with open(r'model/scaler.pkl', 'rb') as f:

                scaler\_loaded = pickle.load(file)

            model\_file\_path = "D:\\copper project\\t.pkl"

            with open(model\_file\_path, 'rb') as file:

            #with open(r'model/scaler.pkl', 'rb') as f:

                t\_loaded = pickle.load(file)

            model\_file\_path = "D:\\copper project\\/s.pkl"

            with open(model\_file\_path, 'rb') as file:

            #with open(r'model/scaler.pkl', 'rb') as f:

                s\_loaded = pickle.load(file)

            new\_sample= np.array([[np.log(float(quantity\_tons)),application,np.log(float(thickness)),float(width),country,float(customer),int(product\_ref),item\_type,status]])

            new\_sample\_ohe = t\_loaded.transform(new\_sample[:, [7]]).toarray()

            new\_sample\_be = s\_loaded.transform(new\_sample[:, [8]]).toarray()

            new\_sample = np.concatenate((new\_sample[:, [0,1,2, 3, 4, 5, 6,]], new\_sample\_ohe, new\_sample\_be), axis=1)

            new\_sample1 = scaler\_loaded.transform(new\_sample)

            new\_pred = loaded\_model.predict(new\_sample1)[0]

            st.write('## :green[Predicted selling price:] ', np.exp(new\_pred))

with tab2:

        with st.form("my\_form1"):

            col1,col2,col3=st.columns([5,1,5])

            with col1:

                cquantity\_tons = st.text\_input("Enter Quantity Tons (Min:611728 & Max:1722207579)")

                cthickness = st.text\_input("Enter thickness (Min:0.18 & Max:400)")

                cwidth = st.text\_input("Enter width (Min:1, Max:2990)")

                ccustomer = st.text\_input("customer ID (Min:12458, Max:30408185)")

                cselling = st.text\_input("Selling Price (Min:1, Max:100001015)")

            with col3:

                st.write(' ')

                citem\_type = st.selectbox("Item Type", item\_type\_options,key=21)

                ccountry = st.selectbox("Country", sorted(country\_options),key=31)

                capplication = st.selectbox("Application", sorted(application\_options),key=41)

                cproduct\_ref = st.selectbox("Product Reference", product,key=51)

                csubmit\_button = st.form\_submit\_button(label="PREDICT STATUS")

            cflag=0

            pattern = "^(?:\d+|\d\*\.\d+)$"

            for k in [cquantity\_tons,cthickness,cwidth,ccustomer,cselling]:

                if re.match(pattern, k):

                    pass

                else:

                    cflag=1

                    break

        if csubmit\_button and cflag==1:

            if len(k)==0:

                st.write("please enter a valid number space not allowed")

            else:

                st.write("You have entered an invalid value: ",k)

        if csubmit\_button and cflag==0:

            import pickle

            model\_file\_path = "D:\\copper project\\cmodel.pkl"

            with open(model\_file\_path, 'rb') as file:

            #with open(r'model/scaler.pkl', 'rb') as f:

                cloaded\_model = pickle.load(file)

            model\_file\_path = "D:\\copper project\\cscaler.pkl"

            with open(model\_file\_path, 'rb') as file:

            #with open(r'model/scaler.pkl', 'rb') as f:

                cscaler\_loaded = pickle.load(file)

            model\_file\_path = "D:\\copper project\\ct.pkl"

            with open(model\_file\_path, 'rb') as file:

            #with open(r'model/scaler.pkl', 'rb') as f:

                ct\_loaded = pickle.load(file)

            # Predict the status for a new sample

            # 'quantity tons\_log', 'selling\_price\_log','application', 'thickness\_log', 'width','country','customer','product\_ref']].values, X\_ohe

            new\_sample = np.array([[np.log(float(cquantity\_tons)), np.log(float(cselling)), capplication, np.log(float(cthickness)),float(cwidth),ccountry,int(ccustomer),int(product\_ref),citem\_type]])

            new\_sample\_ohe = ct\_loaded.transform(new\_sample[:, [8]]).toarray()

            new\_sample = np.concatenate((new\_sample[:, [0,1,2, 3, 4, 5, 6,7]], new\_sample\_ohe), axis=1)

            new\_sample = cscaler\_loaded.transform(new\_sample)

            new\_pred = cloaded\_model.predict(new\_sample)

            #st.write(new\_pred)

            if new\_pred.all()==1:

                st.write('## :green[The Status is Won] ')

            else:

                st.write('## :red[The status is Lost] ')

import datetime

current\_date\_time = datetime.datetime.now()

st.write(f'<h6 style="color:black;">Kesavan sekar DT1819 | {current\_date\_time}</h6>', unsafe\_allow\_html=True)

import streamlit as st

def main():

    #st.title("Kesavan sekar DT1819")

    # Define image URLs

    images = [

        "https://tse2.mm.bing.net/th?id=OIP.hAojpTfiSG\_elgZ8QPdKeAHaDt&pid=Api&P=0&h=220",

        "https://tse4.explicit.bing.net/th?id=OIP.akghOiuTLD5tuUc-sE0n6QHaE7&pid=Api&P=0&h=220",

        "https://www.rkmi.co.in/images/commerical-copper-coil2.jpg",

        "https://4.imimg.com/data4/EH/WH/MY-2772533/industrial-copper-tubes-500x500.jpg",

        "https://public.blenderkit.com/thumbnails/assets/2d412869cca14739a7a42659cb9b2058/files/thumbnail\_dd3d74f6-732f-4d5a-b6d8-0848c345a207.png.512x512\_q85.png",

        "https://5.imimg.com/data5/SELLER/Default/2021/10/IP/RQ/TF/75031140/industrial-copper-tube-1000x1000.jpg",

    ]

    # Create HTML for scrolling images

    html = """

    <style>

    #footer {

        width: 100%;

        overflow: hidden;

        position: fixed;

        bottom: 0;

        background-color: #f1f1f1;

        white-space: nowrap;

        height: 150px;

    }

    .scrolling-wrapper {

        animation: scroll 30s linear infinite;

    }

    @keyframes scroll {

        0% { transform: translateX(100%); }

        100% { transform: translateX(-100%); }

    }

    .image-wrapper {

        display: inline-block;

        margin: 0 20px;

    }

    .image-wrapper img {

        width: 150px; /\* Set width of image \*/

        height: auto; /\* Maintain aspect ratio \*/

    }

    </style>

    <div id="footer">

        <div class="scrolling-wrapper">

            """

    for image in images:

        html += f'<div class="image-wrapper"><img src="{image}" alt="image"></div>'

    html += """

        </div>

    </div>

    """

    # Display scrolling images using st.markdown

    st.markdown(html, unsafe\_allow\_html=True)

if \_\_name\_\_ == "\_\_main\_\_":

    main()