import streamlit as st

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

import seaborn as sns

import time

import io

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

#regression models

from sklearn.linear\_model import LinearRegression

from sklearn.svm import SVR

from sklearn.ensemble import RandomForestRegressor

from sklearn.ensemble import GradientBoostingRegressor

from sklearn import metrics

import joblib

# st.table(df)

df1 = 0

gr = GradientBoostingRegressor()

rad =st.sidebar.radio("Navigation",["Home", "Prediction", "Contribute", "About Us"])

if rad == "Home":

    st.markdown("<center><h3>WELCOME</h3></center>", True)

    st.markdown("<h1><center>Health Insurance Cost Prediction</center></h1>", True)

    st.image("health-insurance-concept.jpg",width=800)

    data = pd.read\_csv("D:\Python\Major Project\health insurance.csv")

    st.markdown("<h4>Dataset :</h4>", True)

    if st.checkbox("Show Dataset: "):

        st.dataframe(data,width=1000,height=600)

    df = pd.DataFrame(data = data)

    #age

    sns.set()

    fig = plt.figure(figsize=(5,5))

    sns.distplot(data['age'])

    plt.title('Age Distribution')

    # plt.show()

    st.pyplot(fig)

    #gender

    fig = plt.figure(figsize=(5,5))

    sns.countplot(x='sex', data = data)

    plt.title('Sex Distribution')

    # plt.show()

    st.pyplot(fig)

    st.write(data['sex'].value\_counts())

    #bmi

    sns.set()

    fig = plt.figure(figsize=(5,5))

    sns.distplot(data['bmi'])

    plt.title('BMI Distribution')

    # plt.show()

    st.pyplot(fig)

    #children

    fig = plt.figure(figsize=(5,5))

    sns.countplot(x='children', data = data)

    plt.title('Children')

    st.pyplot(fig)

    st.write(data['children'].value\_counts())

    #smoker

    fig = plt.figure(figsize=(5,5))

    sns.countplot(x='smoker', data = data)

    plt.title('Smoker')

    st.pyplot(fig)

    st.write(data['smoker'].value\_counts())

    #regionn

    fig = plt.figure(figsize=(5,5))

    sns.countplot(x='region', data = data)

    plt.title('Region')

    st.pyplot(fig)

    st.write(data['region'].value\_counts())

    #charges

    fig = plt.figure(figsize=(5,5))

    sns.distplot(data['charges'])

    plt.title('Charges')

    st.pyplot(fig)

    st.write(data['charges'].value\_counts())

    st.markdown("<h4>Shape of Dataset: </h4>",True)

    st.write(data.shape)

    st.markdown("<h4>Number of Rows: </h4>", True)

    st.write(data.shape[0])

    st.markdown("<h4>Number of Columns: </h4>", True)

    st.write(data.shape[1])

    st.markdown("<h4>Information about data: </h4>", True)

    buffer = io.StringIO()

    data.info(buf=buffer)

    st.text(buffer.getvalue())

    # st.text(s)

    st.markdown("<h4>Statistics of data: </h4>", True)

    st.write(data.describe(include='all'))

    st.markdown("<h4>Converting string values of Columns to numerical values: </h4>", True)

    st.write("Before converting: ", data.head())

    data['sex'] = data['sex'].map({'female':0,'male':1})

    data['smoker'] = data['smoker'].map({'no':0,'yes':1})

    data['region'] = data['region'].map({'southwest':1,'southeast':2, 'northwest':3, 'northeast':4})

    st.write("After converting: ", data.head())

    st.markdown("<h4>Select a Column</h4>", True)

    col = st.multiselect('',data.columns)

    fig = plt.figure(figsize = (12, 6))

    try:

        plots = st.radio(

            "Which plot",

            ('Line Plot', 'Box Plot'))

        if plots == 'Line Plot':

            plt.plot(data[col])

        else:

            plt.boxplot(data[col])

        plt.show()

    except:

        pass

    plt.show()

    st.pyplot(fig)

    #separating independent and dependent variable

    X = data.drop(['charges'], axis=1)

    y = data['charges']

    # st.write(X)

    # st.write(y)

    #train test split

    X\_train, X\_test,y\_train,y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=45)

    #displaying X\_train and y\_train

    st.markdown("<h4>X\_Train: </h4>", True)

    st.write(X\_train)

    st.markdown("<h4>y\_Train: </h4>", True)

    st.write(y\_train)

    #Training Model

    st.markdown("<h4>Models used for Model training: </h4>", True)

    lr = LinearRegression()

    lr.fit(X\_train, y\_train)

    st.write(lr)

    svm = SVR()

    svm.fit(X\_train, y\_train)

    st.write(svm)

    rf = RandomForestRegressor()

    rf.fit(X\_train, y\_train)

    st.write(rf)

    gr = GradientBoostingRegressor()

    gr.fit(X\_train, y\_train)

    st.write(gr)

    #Prediction

    st.markdown("<h4>Prediction: </h4>", True)

    y\_pred1 = lr.predict(X\_test)

    st.write("Predicted value of Linear Regression Model: ", y\_pred1)

    y\_pred2 = svm.predict(X\_test)

    st.write("Predicted value of SVR Model: ", y\_pred2)

    y\_pred3 = rf.predict(X\_test)

    st.write("Predicted value of RandomForestRegressor Model: ", y\_pred3)

    y\_pred4 = gr.predict(X\_test)

    st.write("Predicted value of GradientBoostingRegressor Model: ", y\_pred4)

    st.markdown("<h4>Comparison: </h4>", True)

    df1 = pd.DataFrame({'Actual': y\_test, 'Linear Regression': y\_pred1, 'SVR': y\_pred2, 'Random Forest': y\_pred3, 'Gradient Boosting': y\_pred4})

    st.write(df1)

    #comparing performance visually

    plt.subplot(221) #2 rows 2 columns and 2x2

    plt.plot(df1['Actual'].iloc[0:11], label = "Actual")

    plt.plot(df1['Linear Regression'].iloc[0:11], label = "Linear Regression")

    plt.legend()

    plt.subplot(222)

    plt.plot(df1['Actual'].iloc[0:11], label = "Actual")

    plt.plot(df1['SVR'].iloc[0:11], label = "SVR")

    plt.legend()

    plt.subplot(223)

    plt.plot(df1['Actual'].iloc[0:11], label = "Actual")

    plt.plot(df1['Random Forest'].iloc[0:11], label = "Random Forest")

    plt.legend()

    plt.subplot(224)

    plt.plot(df1['Actual'].iloc[0:11], label = "Actual")

    plt.plot(df1['Gradient Boosting'].iloc[0:11], label = "Gradient Boosting")

    plt.legend()

    plt.tight\_layout()

    st.pyplot(plt)

    #evaluating model using r square

    score1 = metrics.r2\_score(y\_test,y\_pred1)

    score2 = metrics.r2\_score(y\_test,y\_pred2)

    score3 = metrics.r2\_score(y\_test,y\_pred3)

    score4 = metrics.r2\_score(y\_test,y\_pred4)

    st.markdown("<h4>R square values of each model: </h4>", True)

    st.write("Linear Regression: ",score1)

    st.write("SVR: ",score2)

    st.write("Random Forestx: ",score3)

    st.write("Gradient Boosting: ",score4)

    #mean absolute error

    mae1 = metrics.mean\_absolute\_error(y\_test, y\_pred1)

    mae2 = metrics.mean\_absolute\_error(y\_test, y\_pred2)

    mae3 = metrics.mean\_absolute\_error(y\_test, y\_pred3)

    mae4 = metrics.mean\_absolute\_error(y\_test, y\_pred4)

    st.markdown("<h4>Mean Absolute Error values of each model: </h4>", True)

    st.write("Linear Regression: ",mae1)

    st.write("SVR: ",mae2)

    st.write("Random Forestx: ",mae3)

    st.write("Gradient Boosting: ",mae4)

if rad == "About Us":

    progress = st.progress(0)

    for i in range(100):

        time.sleep(0.02)

        progress.progress(i+1)

    st.snow()

    st.title("About Us")

    st.markdown(

            """

            <style>

            .container {

                display: flex;

                justify-content: space-around;

                align-items: center;

                margin-bottom: 50px;

            }

            .person {

                text-align: center;

                padding: 20px;

            }

            .person h3 {

                margin-bottom: 10px;

            }

            .person p {

                font-size: 18px;

                line-height: 1.5;

            }

            </style>

            """,True)

    st.markdown(

            """

            <div class="container">

                <div class="person">

                    <h3>Khushil Bhimani</h3>

                        <h5>B.E in Computer Engineering<br>

                        College Name: Vidyalankar Institute of Technology</h5>

                    <strong>Contact Information:</strong>

                    <ul>

                        <li>Email: khushilbhimani06@gmail.com.com</li>

                        <li>Phone No.:9324130035</li>

                        </ul>

                        </div>

            """,True)

    st.markdown(

            """

            <div class="container">

                <div class="person">

                    <h3>Harshal Bolake</h3>

                    <h5>B.E in Computer Engineering<br>

                        College Name: Terna Engineering College</h5>

                    <strong>Contact Information:</strong>

                    <ul>

                        <li>Email: harshalb321@gmail.com</li>

                        <li>Phone No.:9867701376</li>

                    </ul>

                </div>

            """,True)

if rad == "Prediction":

    # plt.style.use('dark\_background')

    data = pd.read\_csv("D:\Python\Major Project\health insurance.csv")

    data['sex'] = data['sex'].map({'female':0,'male':1})

    data['smoker'] = data['smoker'].map({'no':0,'yes':1})

    data['region'] = data['region'].map({'southwest':1,'southeast':2, 'northwest':3, 'northeast':4})

    gr = GradientBoostingRegressor()

    X = data.drop(['charges'], axis=1)

    # st.write(X)

    y = data['charges']

    gr.fit(X,y)

    # joblib.dump(gr,'model\_train')

    # model = joblib.load('model\_train')

    # model.predict()

    st.title("Health Insurance Predictor")

    age = st.number\_input('Enter your age ')

    gender = st.radio(

    "What\'s your gender",

    ('Male', 'Female'))

    if gender == 'Male':

        gen = 1.0

    else:

        gen = 0.0

    #sns.set()

    bmi = st.number\_input('BMI: ')

    children = st.number\_input('Number of children ')

    smoker = st.radio(

    "Are you smoker?",

    ('Yes', 'No'))

    if smoker == 'Yes':

        smoke = 1.0

    else:

        smoke = 0.0

    region = st.radio(

    "Select your region",

    ('SouthWest', 'SouthEast', 'NorthWest', 'NorthEast'))

    if region == 'SouthWest':

        reg = 1.0

    elif region == 'SouthEast':

        reg = 2.0

    elif region == 'NorthWest':

        reg = 3.0

    else:

        reg = 4.0

    if st.button('PREDICT'):

        joblib.dump(gr,'model\_train') #training model using joblib

        model = joblib.load('model\_train')

        result  = model.predict([[age, gen, bmi, children, smoke, reg]])

        st.success(f'Insurance Cost: $ {result[0]}')

        # st.write('Insurance Cost: $', result[0])

    else:

        st.error('Some Error Occurred', icon="☠️")

if rad == "Contribute":

    st.markdown("<h2>Contribute Your Experience </h2>",  True)

    age = st.number\_input('Enter your age: ')

    gender = st.radio(

    "What\'s your gender",

    ('Male', 'Female'))

    if gender == 'Male':

        gen = 'male'

    else:

        gen = 'female'

    #sns.set()

    bmi = st.number\_input('BMI: ')

    children = st.number\_input('Number of children ')

    smoker = st.radio(

    "Are you smoker?",

    ('Yes', 'No'))

    if smoker == 'Yes':

        smoke = 'yes'

    else:

        smoke = 'no'

    region = st.radio(

    "Select your region",

    ('SouthWest', 'SouthEast', 'NorthWest', 'NorthEast'))

    if region == 'SouthWest':

        reg = 'southwest'

    elif region == 'SouthEast':

        reg = 'southeast'

    elif region == 'NorthWest':

        reg = 'northwest'

    else:

        reg = 'northeast'

    charges = st.number\_input('Charges($)')

    #submit button

    if st.button('Submit'):

        to\_add = {'age':[age], 'sex':[gen], 'bmi':[bmi], 'children':[children], 'smoker':[smoke], 'region':[reg], 'charges ':[charges]}

        to\_add = pd.DataFrame(to\_add)

        to\_add.to\_csv("D://Python//Major Project//health insurance.csv", mode='a', header = False, index = False)

        st.success("Submitted!")

        st.markdown("<h3>Thanks for sharing.</h3>",True)

    else:

        st.markdown("<h3>Can you share your experience.</h3>",True)