GUI

Importing Packages

In [7]:
```
import tkinter as tk
from tkinter import ttk
import pickle
import joblib
import numpy as np
import seaborn as sns
from sklearn.utils import shuffle
from sklearn.linear_model import LogisticRegression
```

Basics of tkinter

Creating a Window

In [8]:
```
window=tk.Tk()
tk.mainloop()
```

Creating Label widget in window

In [3]:
```
window=tk.Tk()
label1=tk.Label(window,text="Hello")
label1.pack()
tk.mainloop()
```

Creating Entry widget in window

In [4]:
```
window=tk.Tk()
x1=tk.StringVar()
tk.Entry(window,textvariable=x1).pack()
tk.mainloop()
print(x1.get())
```

```
Hello
```

Creating Button widget in window

In [5]:
```
window=tk.Tk()
tk.Button(window,text="click",command= lambda : print("Clicked")).pack()
tk.mainloop()
```

```
Clicked
Clicked
```

Working on Iris Dataset

In [9]:
```
# Importing Dataset and printing head
iris=sns.load_dataset("iris")
iris.head()
```

Out[9]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

In [10]:
```
# unique species
iris.species.unique()
```

Out[10]:
```
array(['setosa', 'versicolor', 'virginica'], dtype=object)
```

In [11]:
```
# replacing the name of species with number
num=[]
for i in iris.species:
    if i == "setosa":
        num.append(0)
    elif i=='versicolor':
        num.append(1)
    elif i=='virginica':
        num.append(2)
    else:
        pass
iris['species_num']=num
iris.head()
```

Out[11]:

	sepal_length	sepal_width	petal_length	petal_width	species	species_num
0	5.1	3.5	1.4	0.2	setosa	0
1	4.9	3.0	1.4	0.2	setosa	0
2	4.7	3.2	1.3	0.2	setosa	0
3	4.6	3.1	1.5	0.2	setosa	0
4	5.0	3.6	1.4	0.2	setosa	0

In [12]:
```
iris.species_num.unique()
```

Out[12]:
```
array([0, 1, 2], dtype=int64)
```

In [13]:
```
iris.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object 
 5   species_num   150 non-null    int64  
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

In [14]:
```
# dropping the species name column
iris.drop('species',axis=1,inplace=True)
iris.head()
```

Out[14]:

	sepal_length	sepal_width	petal_length	petal_width	species_num
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

In [15]:
```
# shuffling data
iris=shuffle(iris)
iris.reset_index(drop=True)
iris.head()
```

Out[15]:

	sepal_length	sepal_width	petal_length	petal_width	species_num
3	4.6	3.1	1.5	0.2	0
68	6.2	2.2	4.5	1.5	1
125	7.2	3.2	6.0	1.8	2
38	4.4	3.0	1.3	0.2	0
2	4.7	3.2	1.3	0.2	0

In [16]:
```
iris.shape
```

Out[16]: (150, 5)

In [17]:
```
# Splitting into train and test
train=np.array(iris[:120])
test=np.array(iris[120:])
print(train.shape,test.shape)
```

```
(120, 5) (30, 5)
```

In [18]:
```
# splitting training and testing in x_train, y_train, x_test and y_test
x_train,y_train=train[:,:-1],train[:,-1]
x_test,y_test=test[:,:-1],test[:,-1]
print(x_train.shape,y_train.shape,x_test.shape,y_test.shape)
```

```
(120, 4) (120,) (30, 4) (30,)
```

In [19]:
```
# training model
lr=LogisticRegression()
lr.fit(X=x_train,y=y_train)
```

Out[19]: LogisticRegression()

In [20]:
```
lr.score(x_test,y_test)
```

Out[20]: 0.9666666666666667

Storing using pickle method

In [21]:
```
Pkl_Filename = "Pickle_Model.pkl"
with open(Pkl_Filename, 'wb') as file:
    pickle.dump(lr, file)
```

In [22]:
```
with open(Pkl_Filename, 'rb') as file:
    Pickled_LR_Model = pickle.load(file)
Pickled_LR_Model
```

Out[22]:
```
LogisticRegression()
```

In [23]:
```
Pickled_LR_Model.score(x_test,y_test)
```

Out[23]:
```
0.9666666666666667
```

Storing using Joblib method

In [24]:
```
joblib_file = "joblib_Model.pkl"
joblib.dump(lr, joblib_file)
```

Out[24]:
```
['joblib_Model.pkl']
```

In [25]:
```
joblib_LR_model = joblib.load(joblib_file)
joblib_LR_model
```

Out[25]:
```
LogisticRegression()
```

In [26]:
```
joblib_LR_model.score(x_test,y_test)
```

Out[26]:
```
0.9666666666666667
```

GUI Tool to predict values

In [29]:

```
path_of_file=""
path_of_variables=""
parameters1=["sepal_length","sepal_width","petal_length","petal_width"]
created_model=1
type_of_file1=""
parameter1_values=[]

def find_label(a):
    if a==0:
        return "setosa"
    elif a==1:
        return 'versicolor'
    elif a==2:
        return 'virginica'

def set_path_of_file():
    global path_of_file
    path_of_file=x1.get()
    print(x1.get())

def set_parameter1_values():
    global parameters1,parameter1_values
    parameter1_values=list(map(float,x4.get().split(",")))

def set_file():
    global type_of_file1
    type_of_file1=x3.get()
    load_model()

def load_model():
    global created_model,path_of_file,type_of_file1
    if type_of_file1=="Pickle":
        with open(path_of_file, 'rb') as fil:
            model = pickle.load(fil)
    elif type_of_file1=="Joblib":
        created_model=joblib.load(path_of_file)

def predict_output():
    global created_model,parameter1_values
    output=list(map(find_label,created_model.predict([parameter1_values])))
    window=tk.Tk()
    window.geometry('200x200')
    label=tk.Label(window,text="The output is: "+str(output),font=("Bold",10))
    label.pack()
    label.config(wraplength=150)
    tk.mainloop()

window=tk.Tk()
window.title("Predictor")
x1=tk.StringVar()
x2=tk.StringVar()
x3=tk.StringVar()
x4=tk.StringVar()
tag=tk.StringVar()
tag.set("  ".join(parameters1))

ttk.Label(window,text="Enter the path of file (.pkl)").grid(row=0,column=0)
ttk.Entry(window,textvariable=x1).grid(row=0,column=1)
ttk.Button(window,text="Submit",command=lambda:set_path_of_file()).grid(row=0,column=2)

ttk.Label(window,text="Enter the type of method").grid(row=1,column=0)
ttk.OptionMenu(window,x3,"Select","Pickle","Joblib").grid(row=1,column=1)
ttk.Button(window,text="submit",command=lambda : set_file()).grid(row=1,column=2)

ttk.Label(window,text="Enter the values of parameters shown below in the same order, separated by ',
ttk.Label(window,text=tag.get()).grid(row=3,column=0,columnspan=3)
ttk.Entry(window,textvariable=x4).grid(row=4,column=0,columnspan=3)
ttk.Button(window,text="Submit",command=lambda:set_parameter1_values()).grid(row=5,column=0,columnsp

ttk.Button(window,text="Predict Output",command=lambda : predict_output()).grid(row=6,column=0,colum

tk.mainloop()
```

Modified GUI

In [3]:

```
path_of_file=""
path_of_variables=""
parameters1=["sepal_length","sepal_width","petal_length","petal_width"]
created_model=1
type_of_file1=""
parameter1_values=[]

def find_label(a):
    if a==0:
        return "setosa"
    elif a==1:
        return 'versicolor'
    elif a==2:
        return 'virginica'

def set_path_of_file():
    global path_of_file
    path_of_file=x1.get()
    print(x1.get())

def set_parameter1_values():
    global parameters1,parameter1_values
    parameter1_values=[float(x_0.get()),float(x_1.get()),float(x_2.get()),float(x_3.get())]

def set_file():
    global type_of_file1
    type_of_file1=x3.get()
    load_model()

def load_model():
    global created_model,path_of_file,type_of_file1
    if type_of_file1=="Pickle":
        with open(path_of_file, 'rb') as fil:
            model = pickle.load(fil)
    elif type_of_file1=="Joblib":
        created_model=joblib.load(path_of_file)

def predict_output():
    global created_model,parameter1_values
    output=list(map(find_label,created_model.predict([parameter1_values])))
    window=tk.Tk()
    window.geometry('200x200')
    label=tk.Label(window,text="The output is: "+str(output),font=("Bold",10))
    label.pack()
    label.config(wraplength=150)
    tk.mainloop()

window=tk.Tk()
window.title("Predictor")
x1=tk.StringVar()
x2=tk.StringVar()
x3=tk.StringVar()
x_0=tk.StringVar()
x_1=tk.StringVar()
x_2=tk.StringVar()
x_3=tk.StringVar()
var={"x_0":x_0,"x_1":x_1,"x_2":x_2,"x_3":x_3}
tag=tk.StringVar()
tag.set("  ".join(parameters1))

ttk.Label(window,text="Enter the path of file (.pkl)").grid(row=0,column=0)
ttk.Entry(window,textvariable=x1).grid(row=0,column=1)
ttk.Button(window,text="Submit",command=lambda:set_path_of_file()).grid(row=0,column=2)

ttk.Label(window,text="Enter the type of method").grid(row=1,column=0)
ttk.OptionMenu(window,x3,"Select","Pickle","Joblib").grid(row=1,column=1)
ttk.Button(window,text="submit",command=lambda : set_file()).grid(row=1,column=2)

ttk.Label(window,text="Enter the values of parameters shown below in the same order, separated by ',
ttk.Label(window,text=tag.get()).grid(row=3,column=0,columnspan=3)
for i in range(len(parameters1)):
    ttk.Label(window,text=parameters1[i]).grid(row=4+i,column=0)
    ttk.Entry(window,textvariable=var["x_"+str(i)]).grid(row=4+i,column=1,columnspan=2)
ttk.Button(window,text="Submit",command=lambda:set_parameter1_values()).grid(row=8,column=0,columnsp
```

```
ttk.Button(window,text="Predict Output",command=lambda : predict_output()).grid(row=9,column=0,colu

tk.mainloop()
```