In this notebook, lets import our models from the converted pickle files.

```
# Importing libraries
import numpy as np
import pandas as pd

# Importing pickle
import pickle
```

Loading the linear regression Model back from pickle file

```
with open('Pickle_reg_Model.pkl', 'rb') as file:
    LR_Model = pickle.load(file)

LR_Model

LinearRegression()

# Linear Regresion

def LinearReg():

    print("Input data for 1st example.")
    rm1 = input("Enter 'rm' value: ")
    lstat1 = input("Enter 'lstat' value: ")
    indus1 = input("Enter 'indus' value: ")
    ptratio1 = input("Enter 'ptratio' value: ")

    print("Input data for 2nd example.")
    rm2 = input("Enter 'rm' value: ")
    lstat2 = input("Enter 'lstat' value: ")
    indus2 = input("Enter 'indus' value: ")
    ptratio2 = input("Enter 'ptratio' value: ")

    list = [[rm1,lstat1,indus1,ptratio1],[rm2,lstat2,indus2,ptratio2]]
    l = np.array(list)
    medv = LR_Model.predict(l)
    print("Medv value of given input data is",medv)
```

Loading the logistic regression Model back from pickle file

```
with open('Pickle_LogReg_Model.pkl', 'rb') as file:
    LogR_Model = pickle.load(file)

LogR_Model

LogisticRegression()

# Logistic Regression

def LogisticReg():

    print("Input data for 1st example.")
```

```
    preg1 = input("No. of times pregnant: ")
    plas1 = input("Plasma glucose concentration a 2 hours in an oral
glucose tolerance test: ")
    pres1 = input("Diastolic BP(in mm Hg): ")
    skin1 = input("Triceps skin fold thickness(in mm): ")
    insu1 = input("2-Hour serum insulin (mu U/ml): ")
    mass1 = input("BMI: ")
    pedi1 = input("Diabetes pedigree function: ")
    age1 = input("Age (years): ")

    print("Input data for 2nd example.")
    preg2 = input("No. of times pregnant: ")
    plas2 = input("Plasma glucose concentration a 2 hours in an oral
glucose tolerance test: ")
    pres2 = input("Diastolic BP(in mm Hg): ")
    skin2 = input("Triceps skin fold thickness(in mm): ")
    insu2 = input("2-Hour serum insulin (mu U/ml): ")
    mass2 = input("BMI: ")
    pedi2 = input("Diabetes pedigree function: ")
    age2 = input("Age (years): ")

    list = [[preg1,plas1,pres1,skin1,insu1,mass1,pedi1,age1],
[preg2,plas2,pres2,skin2,insu2,mass2,pedi2,age2]]
    l = np.array(list)
    outcome = LogR_Model.predict(l)
    print("Outcome of given input examples are ",outcome)
```

Load the KNN Model back from pickle file

```
with open('Pickle_knn_Model.pkl', 'rb') as file:
    knn_Model = pickle.load(file)

knn_Model

KNeighborsClassifier(n_neighbors=7)

def knn():

    print("Input data for 1st example (all in cm).")
    SepalL1 = input("Sepal Length: ")
    SepalW1 = input("Sepal Width: ")
    PetalL1 = input("Petal Length: ")
    PetalW1 = input("Petal Width: ")

    print("Input data for 2nd example (all in cm).")
    SepalL2 = input("Sepal Length: ")
    SepalW2 = input("Sepal Width: ")
    PetalL2 = input("Petal Length: ")
    PetalW2 = input("Petal Width: ")

    list= [[SepalL1,SepalW1,PetalL1,PetalW1],
[SepalL2,SepalW2,PetalL2,PetalW2]]
```

```
    l = np.array(list)
    species = knn_Model.predict(l)
    print("Species of given examples are ",species)
```

Loading the Decision Tree model back from pickle file

```
with open('Pickle_deciTree_Model.pkl', 'rb') as file:
    DT_Model = pickle.load(file)

DT_Model

GridSearchCV(cv=10, estimator=DecisionTreeClassifier(random_state=42),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': array([ 2,  3,  4,  5,  6,  7,
8,  9, 10])},
             scoring='roc_auc')

def DecisionTree():
    print("Input data for 1st example.")
    checAcc1 = input("Checking Acc: ")
    duration1 = input("Duration: ")
    credhis1 = input("Credit History: ")
    amt1 = input("Amount: ")
    savAcc1 = input("Savings Acc: ")
    pes1 = input("Present_emp_since: ")
    ins_rate1 = input("Installment Rate: ")
    ps1 = input("Personal Status: ")
    rs1 = input("Residing since: ")
    age1 = input("Age: ")
    ip1 = input("Inst Plans: ")
    nc1 = input("No. of credits: ")
    j1 = input("Job: ")

    print("Input data for 2nd example.")
    checAcc2 = input("Checking Acc: ")
    duration2 = input("Duration: ")
    credhis2 = input("Credit History: ")
    amt2 = input("Amount: ")
    savAcc2 = input("Savings Acc: ")
    pes2 = input("Present_emp_since: ")
    ins_rate2 = input("Installment Rate: ")
    ps2 = input("Personal Status: ")
    rs2 = input("Residing since: ")
    age2 = input("Age: ")
    ip2 = input("Inst Plans: ")
    nc2 = input("No. of credits: ")
    j2 = input("Job: ")

    list =
[[checAcc1,duration1,credhis1,amt1,savAcc1,pes1,ins_rate1,ps1,rs1,age1
,ip1,nc1,j1],
```

```
[checAcc2,duration2,credhis2,amt2,savAcc2,pes2,ins_rate2,ps2,rs2,age2,
ip2,nc2,j2]]
    l = np.array(list)
    status = DT_Model.predict(l)
    print("Status of given inputs are ",status)
```

Loading the MLP model back from pickle file

```
with open('Pickle_MLP_Model.pkl', 'rb') as file:
    mlp_Model = pickle.load(file)

mlp_Model

MLPClassifier(hidden_layer_sizes=(13, 13, 13), learning_rate_init=0.5,
              max_iter=500)

def MLP():
    print("Input contents for 1st example.")
    Al1 = input("Alcohol: ")
    Ma1 = input("Malic acid: ")
    Ash1 = input("Ash: ")
    AlA1 = input("Alcalinity of ash: ")
    Mg1 = input("Magnesium: ")
    Tp1 = input("Total phenols: ")
    F1 = input("Flavanoids: ")
    Np1 = input("Nonflavanoid phenols: ")
    P1 = input("Proanthocyanins: ")
    Ci1 = input("Color intensity: ")
    Hue1 = input("Hue: ")
    dw1 = input("OD280/OD315 of diluted wines: ")
    pr1 = input("Proline: ")


    print("Input data for 2nd example.")
    Al2 = input("Alcohol: ")
    Ma2 = input("Malic acid: ")
    Ash2 = input("Ash: ")
    AlA2 = input("Alcalinity of ash: ")
    Mg2 = input("Magnesium: ")
    Tp2 = input("Total phenols: ")
    F2 = input("Flavanoids: ")
    Np2 = input("Nonflavanoid phenols: ")
    P2 = input("Proanthocyanins: ")
    Ci2 = input("Color intensity: ")
    Hue2 = input("Hue: ")
    dw2 = input("OD280/OD315 of diluted wines: ")
    pr2 = input("Proline: ")

    list = [[Al1,Ma1,Ash1,AlA1,Mg1,Tp1,F1,Np1,P1,Ci1,Hue1,dw1,pr1],
```

```
            [Al2,Ma2,Ash2,AlA2,Mg2,Tp2,F2,Np2,P2,Ci2,Hue2,dw2,pr2]]
    l = np.array(list)
    Class = DT_Model.predict(l)
    print("Classes of given examples are ",Class)
```

ML Dashboard:

```
# code to let user select an option to predict value of his choice.
while(True):

    print("1 - Prediction of Boston House prices.")
    print("2 - Prediction of Diabetes.")
    print("3 - Classification of specie of flower.")
    print("4 - Classification of German Credit card data.")
    print("5 - Prediction of class of Wine given contents of its
ingredients.")
    print("0 - to exit the loop and stop executing.")
    opt =int(input("Enter your choice: "))

    if(opt == 1):
        LinearReg()
    elif(opt == 2):
        LogisticReg()
    elif(opt == 3):
        knn()
    elif(opt == 4):
        DecisionTree()
    elif(opt == 5):
        MLP()
    elif(opt == 0):
        break
    else:
        print("Invalid choice. Opt again!")
```

```
1 - Prediction of Boston House prices.
2 - Prediction of Diabetes.
3 - Classification of specie of flower.
4 - Classification of German Credit card data.
5 - Prediction of class of Wine given contents of its ingredients.
0 - to exit the loop and stop executing.
Enter your choice: 0
```