```python
# Importing required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split

# Import pickle Package

import pickle
```

1st model - Linear Regression

```python
# Importing the boston.csv dataset
df1 = pd.read_csv("Bosto_housing.csv")
df1.drop('Unnamed: 0',axis=1,inplace=True)
df1.head()
```

	rm	lstat	indus	ptratio	medv
0	6.575	4.98	2.31	15.3	24.0
1	6.421	9.14	7.07	17.8	21.6
2	7.185	4.03	7.07	17.8	34.7
3	6.998	2.94	2.18	18.7	33.4
4	7.147	5.33	2.18	18.7	36.2

```python
# independent & dependent data
X = df1.iloc[:, :-1].values

y = df1.iloc[:, -1].values

# splitting data into trainig & testing data
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.20, random_state=42)

# Creating a linear regression model using the linear_regression
module from the sklearn library
from sklearn.linear_model import LinearRegression
reg = LinearRegression()
reg.fit(X_train,y_train)
```

```
LinearRegression()
```

```python
# Save the LR Model to file in the current working directory

Pkl_Filename = "Pickle_reg_Model.pkl"

with open(Pkl_Filename, 'wb') as file:
    pickle.dump(reg, file)
```

2nd model - Logistic Regression

```
# Importing the Diabetes.csv dataset
df2 = pd.read_csv("diabetes.csv")
df2.head()
```

```
   preg  plas  pres  skin  insu  mass   pedi  age            class
0     6   148    72    35     0  33.6  0.627   50  tested_positive
1     1    85    66    29     0  26.6  0.351   31  tested_negative
2     8   183    64     0     0  23.3  0.672   32  tested_positive
3     1    89    66    23    94  28.1  0.167   21  tested_negative
4     0   137    40    35   168  43.1  2.288   33  tested_positive
```

This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective is to predict based on diagnostic measurements whether a patient has diabetes.

Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage.

Number of Instances: 768

Number of Attributes: 8 plus class

Attributes:

Pregnancies: Number of times pregnant

Glucose: Plasma glucose concentration a 2 hours in an oral glucose tolerance test

BloodPressure: Diastolic blood pressure (mm Hg)

SkinThickness: Triceps skin fold thickness (mm)

Insulin: 2-Hour serum insulin (mu U/ml)

BMI: Body mass index (weight in kg/(height in m)^2)

DiabetesPedigreeFunction: Diabetes pedigree function

Age: Age (years)

Outcome: Class variable (0 or 1)

```
# Assigning the predictor variables to 'x' and response variable to 'y'

x = df2.drop('class',axis=1)
y = df2['class']

# Using the standard scaler function for scaling the data.
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X = sc.fit_transform(x)
```

```
# Splitting the dataset into train and test data.
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test =
train_test_split(X,y,test_size=0.3,random_state=30)

# Using the linear_model module of sklearn to build a Logistic
Regression model.
from sklearn.linear_model import LogisticRegression
logReg = LogisticRegression()
logReg.fit(x_train,y_train)

LogisticRegression()

# Save the LogisticReg Model to file in the current working directory

Pkl_Filename = "Pickle_LogReg_Model.pkl"

with open(Pkl_Filename, 'wb') as file:
    pickle.dump(logReg, file)
```

3rd model - KNN Classifier

```
# Importing the Iris dataset
df3 = pd.read_csv("Iris.csv")
df3.drop('Id',axis=1,inplace =True)
df3.head()

   SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
Species
0            5.1           3.5            1.4           0.2  Iris-
setosa
1            4.9           3.0            1.4           0.2  Iris-
setosa
2            4.7           3.2            1.3           0.2  Iris-
setosa
3            4.6           3.1            1.5           0.2  Iris-
setosa
4            5.0           3.6            1.4           0.2  Iris-
setosa

# Assigning the predictor variables to 'x' and response variable to
'y'

X = df3.iloc[:,:-1]
Y = df3.iloc[:,-1]

X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size =
0.2,random_state = 0)

# Building a KNN Classifier using sklearn library.
```

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=7)
knn.fit(X_train, Y_train)
```

```
KNeighborsClassifier(n_neighbors=7)
```

```
# Save the KNN Model to file in the current working directory

Pkl_Filename = "Pickle_knn_Model.pkl"

with open(Pkl_Filename, 'wb') as file:
    pickle.dump(knn, file)
```

4th Model - Decision Trees

```
# Importing the dataset
df4 = pd.read_csv("German Credit Data.csv")
df4
```

```
    checkin_acc  duration credit_history  amount savings_acc  \
0           A11         6            A34    1169         A65
1           A12        48            A32    5951         A61
2           A14        12            A34    2096         A61
3           A11        42            A32    7882         A61
4           A11        24            A33    4870         A61
..          ...       ...            ...     ...         ...
995         A14        12            A32    1736         A61
996         A11        30            A32    3857         A61
997         A14        12            A32     804         A61
998         A11        45            A32    1845         A61
999         A12        45            A34    4576         A62

    present_emp_since  inst_rate personal_status  residing_since  age
\
0                 A75          4             A93               4   67

1                 A73          2             A92               2   22

2                 A74          2             A93               3   49

3                 A74          2             A93               4   45

4                 A73          3             A93               4   53

..                ...        ...             ...             ...  ...

995               A74          3             A92               4   31

996               A73          4             A91               4   40

997               A75          4             A93               4   38
```

```
998              A73         4            A93              4   23

999              A71         3            A93              4   27


    inst_plans  num_credits   job  status
0         A143            2  A173       0
1         A143            1  A173       1
2         A143            1  A172       0
3         A143            1  A173       0
4         A143            2  A173       1
..         ...          ...   ...     ...
995       A143            1  A172       0
996       A143            1  A174       0
997       A143            1  A173       0
998       A143            1  A173       1
999       A143            1  A173       0

[1000 rows x 14 columns]
```

```
# Select the categorical columns and use Label Encoder for encoding
the categorical values to numerical.
cat_cols = df4.select_dtypes(include = ['object']).copy()
cat_cols.columns

from sklearn.preprocessing import LabelEncoder
for i in cat_cols.columns:
    df4[i] = LabelEncoder().fit_transform(df4[i])

# Splitting predictor and response variables

X = df4.drop(columns = ['status'],axis = 1)
y = df4.status

# splitting the dataset into train and test datasets
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size =
0.2,random_state = 42)

# Building a decision tree model
# Importing decisiontree classifier and gridsearchcv modules from
sklearn

from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier
# configuring the gridsearch again to print all the evaluation metrics

tree_para = {'criterion':['gini','entropy'],'max_depth':np.arange(2,
11)}
grid_search_cv = GridSearchCV(DecisionTreeClassifier(random_state =
```

```
42), tree_para, cv = 10, scoring='roc_auc')
grid_search_cv.fit(X_train,y_train)
```

```
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')
```

```
# Save the DT Model to file in the current working directory

Pkl_Filename = "Pickle_deciTree_Model.pkl"

with open(Pkl_Filename, 'wb') as file:
    pickle.dump(grid_search_cv, file)
```

5th Model - Multi Layer Perceptron (NN)

```
# importing the wine dataset
df5 = pd.read_csv("wine.csv")
df5.head()
```

```
   Class  Alcohol  Malic acid   Ash  Alcalinity of ash  Magnesium  \
0      1    14.23        1.71  2.43               15.6        127
1      1    13.20        1.78  2.14               11.2        100
2      1    13.16        2.36  2.67               18.6        101
3      1    14.37        1.95  2.50               16.8        113
4      1    13.24        2.59  2.87               21.0        118

   Total phenols  Flavanoids  Nonflavanoid phenols  Proanthocyanins  \
0           2.80        3.06                  0.28             2.29
1           2.65        2.76                  0.26             1.28
2           2.80        3.24                  0.30             2.81
3           3.85        3.49                  0.24             2.18
4           2.80        2.69                  0.39             1.82

   Color intensity   Hue  OD280/OD315 of diluted wines  Proline
0             5.64  1.04                          3.92     1065
1             4.38  1.05                          3.40     1050
2             5.68  1.03                          3.17     1185
3             7.80  0.86                          3.45     1480
4             4.32  1.04                          2.93      735
```

Title of Database: Wine recognition data

These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines.

The 13 attributes are

1) Alcohol

2) Malic acid

3) Ash

4) Alcalinity of ash

5) Magnesium

6) Total phenols

7) Flavanoids

8) Nonflavanoid phenols

9) Proanthocyanins

10)Color intensity

11)Hue

12)OD280/OD315 of diluted wines

13)Proline

Number of Instances

class 1 59

class 2 71

class 3 48

```
#Convert variable to categorical.
df5.Class=df5.Class.astype('category')

#Splitting the predictor & response variables
X = df5.drop(columns = ['Class'],axis =1)
y = df5.Class

from sklearn.preprocessing import StandardScaler
X = StandardScaler().fit_transform(X)

# Spliting of the dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X,y,
test_size=0.3,random_state = 42)

# Importing Multi-Layer Perceptron Classifier model
from sklearn.neural_network import MLPClassifier

#Create a MLP Classifier with 3 layers
model = MLPClassifier(hidden_layer_sizes=(13,13,13),
activation='relu', max_iter=500,learning_rate_init=0.5)
```

```
#Train the model using the training sets
model.fit(X_train, y_train)

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

# Save the Modle to file in the current working directory

Pkl_Filename = "Pickle_MLP_Model.pkl"

with open(Pkl_Filename, 'wb') as file:
    pickle.dump(model, file)
```