**Random Forest Classifier:**

import pandas as pd

import numpy as np

from pandas import Series, DataFrame

from sklearn.cross\_validation import train\_test\_split

from sklearn.neighbors import KNeighborsClassifier

from sklearn.cross\_validation import cross\_val\_score

from sklearn.ensemble import RandomForestClassifier

import matplotlib as plt

% matplotlib inline

PLI\_train = DataFrame(pd.read\_csv("train.csv"))

PLI\_test = DataFrame(pd.read\_csv("test.csv"))

PLI\_test\_1 = DataFrame(pd.read\_csv("test.csv"))

PLI\_train = PLI\_train.fillna(PLI\_train.median())

PLI\_test = PLI\_test.fillna(PLI\_test.median())

y\_train = PLI\_train["Response"]

PLI\_train = PLI\_train.drop(['Response','Id','Product\_Info\_2'], axis = 1)

PLI\_test = PLI\_test.drop(['Id','Product\_Info\_2'], axis = 1)

PLI\_train = PLI\_train.astype(float)

PLI\_test = PLI\_test.astype(float)

x\_train = PLI\_train

test = PLI\_test

X\_train,X\_test,Y\_train,Y\_test = train\_test\_split(x\_train,y\_train)

PLI\_RF\_1 = RandomForestClassifier(n\_estimators = 10, criterion = 'gini')

scores = cross\_val\_score(PLI\_RF\_1, X\_train, Y\_train, cv = 10, scoring = 'accuracy')

print(scores)

print(scores.mean())

PLI\_RF\_2 = RandomForestClassifier(n\_estimators = 10, criterion = 'gini', bootstrap = False)

scores = cross\_val\_score(PLI\_RF\_2, X\_train, Y\_train, cv = 10, scoring = 'accuracy')

print(scores)

print(scores.mean())

PLI\_RF\_3 = RandomForestClassifier(n\_estimators = 10, criterion = 'gini', max\_features = None)

scores = cross\_val\_score(PLI\_RF\_3, X\_train, Y\_train, cv = 10, scoring = 'accuracy')

print(scores)

print(scores.mean())

PLI\_RF\_4 = RandomForestClassifier(n\_estimators = 10, criterion = 'gini', max\_features = 'log2')

scores = cross\_val\_score(PLI\_RF\_4, X\_train, Y\_train, cv = 10, scoring = 'accuracy')

print(scores)

print(scores.mean())

PLI\_RF\_5 = RandomForestClassifier(n\_estimators = 10, criterion = 'gini', max\_features = None, bootstrap = False)

scores = cross\_val\_score(PLI\_RF\_5, X\_train, Y\_train, cv = 10, scoring = 'accuracy')

print(scores)

print(scores.mean())

PLI\_RF\_6 = RandomForestClassifier(n\_estimators = 200, criterion = 'gini')

scores = cross\_val\_score(PLI\_RF\_6, X\_train, Y\_train, cv = 10, scoring = 'accuracy')

print(scores)

print(scores.mean())

PLI\_RF\_main = PLI\_RF\_7.fit(x\_train, y\_train)

Prediction = PLI\_RF\_main.predict(test)

Response = Prediction

Result1 = pd.DataFrame()

ID = PLI\_test\_1['Id']

Result1['Id'] = ID

Result1['Response'] = Response

Result1.head(4)

Result1.to\_csv('Result1.csv', index = False)

**KNN Classifier:**

import pandas as pd

import numpy as np

from pandas import Series, DataFrame

from sklearn.cross\_validation import train\_test\_split

from sklearn.neighbors import KNeighborsClassifier

from sklearn.cross\_validation import cross\_val\_score

import matplotlib as plt

% matplotlib inline

PLI\_train = DataFrame(pd.read\_csv("train.csv"))

PLI\_test = DataFrame(pd.read\_csv("test.csv"))

PLI\_test\_1 = DataFrame(pd.read\_csv("test.csv"))

PLI\_train = PLI\_train.fillna(PLI\_train.median())

PLI\_test = PLI\_test.fillna(PLI\_test.median())

y\_train = PLI\_train["Response"]

PLI\_train = PLI\_train.drop(['Response','Id','Product\_Info\_2'], axis = 1)

PLI\_test = PLI\_test.drop(['Id','Product\_Info\_2'], axis = 1)

PLI\_train = PLI\_train.astype(float)

PLI\_test = PLI\_test.astype(float)

x\_train = PLI\_train

test = PLI\_test

X\_train,X\_test,Y\_train,Y\_test = train\_test\_split(x\_train,y\_train)

KNN = KNeighborsClassifier(n\_neighbors = 5)

scores = cross\_val\_score(KNN, X\_train, Y\_train, cv = 10, scoring = 'accuracy')

print(scores)

print(scores.mean())

k\_range = range(1,5)

k\_scores = []

for k in k\_range:

KNN = KNeighborsClassifier(n\_neighbors = 5)

scores = cross\_val\_score(KNN, X\_train, Y\_train, cv= 10, scoring = 'accuracy')

k\_scores.append(scores.mean())

print(k\_scores)

KNN\_2 = KNeighborsClassifier(n\_neighbors = 10)

scores = cross\_val\_score(KNN\_2, X\_train, Y\_train, cv = 10, scoring = 'accuracy')

print(scores)

print(scores.mean())

KNN\_3 = KNeighborsClassifier(n\_neighbors = 50)

scores = cross\_val\_score(KNN\_3, X\_train, Y\_train, cv = 10, scoring = 'accuracy')

print(scores)

print(scores.mean())

KNN\_4 = KNeighborsClassifier(n\_neighbors = 200)

scores = cross\_val\_score(KNN\_4, X\_train, Y\_train, cv = 10, scoring = 'accuracy')

print(scores)

print(scores.mean())

KNN\_5 = KNeighborsClassifier(n\_neighbors = 75, algorithm = 'brute')

scores = cross\_val\_score(KNN\_5, X\_train, Y\_train, cv = 10, scoring = 'accuracy')

print(scores)

print(scores.mean())

KNN\_6 = KNeighborsClassifier(n\_neighbors = 100, algorithm = 'brute')

scores = cross\_val\_score(KNN\_6, X\_train, Y\_train, cv = 10, scoring = 'accuracy')

print(scores)

print(scores.mean())

KNN\_6.fit(x\_train, y\_train)

Predicted = KNN\_6.predict(test)

Similarly do the further steps as did in previous steps.

**Logistic Regression Classifier:**

import pandas as pd

import numpy as np

from pandas import Series, DataFrame

from sklearn.cross\_validation import train\_test\_split

from sklearn.cross\_validation import cross\_val\_score

from sklearn.linear\_model import LogisticRegression

import matplotlib as plt

% matplotlib inline

PLI\_train = DataFrame(pd.read\_csv("train.csv"))

PLI\_test = DataFrame(pd.read\_csv("test.csv"))

PLI\_test\_1 = DataFrame(pd.read\_csv("test.csv"))

PLI\_train = PLI\_train.fillna(PLI\_train.median())

PLI\_test = PLI\_test.fillna(PLI\_test.median())

y\_train = PLI\_train["Response"]

PLI\_train = PLI\_train.drop(['Response','Id','Product\_Info\_2'], axis = 1)

PLI\_test = PLI\_test.drop(['Id','Product\_Info\_2'], axis = 1)

PLI\_train = PLI\_train.astype(float)

PLI\_test = PLI\_test.astype(float)

x\_train = PLI\_train

test = PLI\_test

X\_train,X\_test,Y\_train,Y\_test = train\_test\_split(x\_train,y\_train)

PLI\_LogR = LogisticRegression(penalty = 'l2', C=1.0)

scores = cross\_val\_score(PLI\_LogR, X\_train, Y\_train, cv = 10, scoring = 'accuracy')

print(scores)

print(scores.mean())

PLI\_LogR\_2 = LogisticRegression(penalty = 'l1', C=1.0, class\_weight = 'balanced')

scores = cross\_val\_score(PLI\_LogR\_2, X\_train, Y\_train, cv = 10, scoring = 'accuracy')

print(scores)

print(scores.mean())

PLI\_LogR\_3 = LogisticRegression(penalty = 'l2', C=0.05)

scores = cross\_val\_score(PLI\_LogR\_3, X\_train, Y\_train, cv = 10, scoring = 'accuracy')

print(scores)

print(scores.mean())

PLI\_LogR\_4 = LogisticRegression(penalty = 'l2', C = 0.01)

scores = cross\_val\_score(PLI\_LogR\_4, X\_train, Y\_train, cv = 10, scoring = 'accuracy')

print(scores)

print(scores.mean())

PLI\_LogR.fit(x\_train, y\_train)

PLI\_LogR.score(x\_train, y\_train)

Similarly do previous steps to save the file for kaggle submission.

**Naïve Bayes Classifier:**

import pandas as pd

import numpy as np

from pandas import Series, DataFrame

from sklearn.cross\_validation import train\_test\_split

from sklearn.cross\_validation import cross\_val\_score

from sklearn.naive\_bayes import GaussianNB

import matplotlib as plt

% matplotlib inline

PLI\_train = DataFrame(pd.read\_csv("train.csv"))

PLI\_test = DataFrame(pd.read\_csv("test.csv"))

PLI\_test\_1 = DataFrame(pd.read\_csv("test.csv"))

PLI\_train = PLI\_train.fillna(PLI\_train.median())

PLI\_test = PLI\_test.fillna(PLI\_test.median())

y\_train = PLI\_train["Response"]

PLI\_train = PLI\_train.drop(['Response','Id','Product\_Info\_2'], axis = 1)

PLI\_test = PLI\_test.drop(['Id','Product\_Info\_2'], axis = 1)

PLI\_train = PLI\_train.astype(float)

PLI\_test = PLI\_test.astype(float)

x\_train = PLI\_train

test = PLI\_test

X\_train,X\_test,Y\_train,Y\_test = train\_test\_split(x\_train,y\_train)

PLI\_GNB = GaussianNB()

scores = cross\_val\_score(PLI\_GNB, X\_train, Y\_train, cv = 10, scoring = 'accuracy')

print(scores)

print(scores.mean())

PLI\_GNB.fit(x\_train, y\_train)

PLI\_GNB.predict(test)