In [2]:
```
import np
import pandas as pd
```

In [3]:
```
x_data = np.load('X2_train_task2.npy', mmap_mode='r')
y = np.load('y2_train_task2.npy', mmap_mode='r')

X_test_task = np.load('X2_test_task2.npy', mmap_mode='r')
```

Modelling

In [4]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test=train_test_split(x_data,y)
```

KNeighborsClassifier

In [5]:
```
from sklearn.neighbors import KNeighborsClassifier

knn=KNeighborsClassifier(n_neighbors=6)
knn.fit(x_train,y_train)

knn_pred=knn.predict(x_test)
```

In [6]:
```
print("KNN accuracy score: ",knn.score(x_test,y_test))
```

```
KNN accuracy score:  0.999398085913204
```

In [ ]:
```
print("train score - " + str(knn.score(x_train,y_train)))
print("test score - " + str(knn.score(x_test, y_test)))
```

Random Forest

In [7]:
```
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier(n_estimators=100,random_state=1)
rf.fit(x_train,y_train)

rf_pred=rf.predict(x_test)

print("RF accuracy score: ",rf.score(x_test,y_test))
```

```
RF accuracy score:  0.9995585963363496
```

In [ ]:
```
print("train score - " + str(rf.score(x_train,y_train)))
print("test score - " + str(rf.score(x_test, y_test)))
```

SVM

In [8]:
```
from sklearn.svm import SVC
svm=SVC(random_state=1,gamma="auto")
svm.fit(x_train,y_train)

svm_pred=svm.predict(x_test)

print("SVM accuracy score: ",svm.score(x_test,y_test))
```

```
SVM accuracy score:  0.9992777030958447
```

In [ ]:
```
print("train score - " + str(svm.score(x_train,y_train)))
print("test score - " + str(svm.score(x_test, y_test)))
```

Decision Tree

In [9]:
```
from sklearn.tree import DecisionTreeClassifier
dt=DecisionTreeClassifier()
dt.fit(x_train,y_train)

dt_pred=dt.predict(x_test)

print("DT accuracy score: ",dt.score(x_test,y_test))
```

```
DT accuracy score:  0.9991573202784856
```

In [ ]:
```
print("train score - " + str(dt.score(x_train,y_train)))
print("test score - " + str(dt.score(x_test, y_test)))
```

Logistic Regression

In [11]:
```
# Training the Logistic Regression model on the Training set

from sklearn.metrics import roc_curve, auc, confusion_matrix, classification_report,accuracy_score
from sklearn.linear_model import LogisticRegression
lg = LogisticRegression(random_state = 0)
lg.fit(x_train, y_train)

#predictin the test result
y_pred_lg = lg.predict(x_test)

print("Logistic Regression accuracy score: ",lg.score(x_test,y_test))
```

```
Logistic Regression accuracy score:  0.9991773840813788
```

In [12]:
```
#calculate accuracy
score_lg = accuracy_score(y_pred_lg,y_test)
score_lg
```

Out[12]:
```
0.9991773840813788
```

In [13]:
```
print("train score - " + str(lg.score(x_train,y_train)))
print("test score - " + str(lg.score(x_test, y_test)))
```

```
train score - 0.9991372564755924
test score - 0.9991773840813788
```

kernal SVM

In [14]:
```
from sklearn.svm import SVC

#fitting kernal SVM to the training set
ksvm = SVC(kernel='rbf', random_state=0 )
ksvm.fit(x_train,y_train)

#predictin the test result
y_pred_ksvm = ksvm.predict(x_test)
```

In [15]:
```
print("kernal SVM accuracy score: ",ksvm.score(x_test,y_test))
```

```
kernal SVM accuracy score:  0.9992777030958447
```

In [16]:
```
#calculate accuracy
score_ksvm = accuracy_score(y_pred_ksvm,y_test)
score_ksvm
```

Out[16]:
```
0.9992777030958447
```

In [ ]:
```
print("train score - " + str(ksvm.score(x_train,y_train)))
print("test score - " + str(ksvm.score(x_test, y_test)))
```

Kernal Navie Bayes

In [17]:
```
from sklearn.naive_bayes import GaussianNB

#fitting kernal Navie bayes to the training set
knb = GaussianNB()
knb.fit(x_train,y_train)


#predictin the test result
y_pred_knb = knb.predict(x_test)
```

In [18]:
```
print("Kernal Navie Bayes accuracy score: ",knb.score(x_test,y_test))
```

```
Kernal Navie Bayes accuracy score:  0.9787123051303144
```

In [19]:
```
#calculate accuracy
score_knb = accuracy_score(y_pred_knb,y_test)
score_knb
```

Out[19]:
```
0.9787123051303144
```

In [ ]:
```
print("train score - " + str(knb.score(x_train,y_train)))
print("test score - " + str(knb.score(x_test, y_test)))
```

In [ ]:
```

```

Creating Submission File

In [21]:
```
# CHOOSE YOUR ALGTHM PREDICT LINE AND PLACE  X_test_task INSIDE ()


# HERE I HAVE CHOOSED Random Forest

rf_pred_test =rf.predict(X_test_task)
```

In [22]:
```
pd.DataFrame({"Id": np.arange(len(rf_pred_test)),
              "Category": rf_pred_test}).astype(int).to_csv( "solution.csv", index=False )
```

In [ ]: