MANOJ KUMAR - 2048015

Regression Models for task3

In [1]:

```
import np
import pandas as pd

from sklearn import metrics
from sklearn.metrics import r2_score,mean_squared_error
```

In [2]:

```
X = np.load('new_X_train3.npy', allow_pickle=True)
Y = np.load('new_y3_train.npy', allow_pickle=True)

X_test_task3 = np.load('new_X_test3.npy', allow_pickle=True)
```

Modelling

In [3]:

```
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_s
```

Simple Linear Regression

In [4]:

```
# from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression
linreg = LinearRegression()
linreg.fit(X_train, y_train)
```

Out[4]:

```
LinearRegression()
```

In [5]:

```
#Training Accuracy
linreg.score(X_train,y_train)

#Prediction
LR_pred=linreg.predict(X_test)
```

In [6]:

```
print("MAE: ",metrics.mean_absolute_error(y_test,LR_pred))
print("MSE: ",metrics.mean_squared_error(y_test,LR_pred))
print("RMSE: ",np.sqrt(metrics.mean_squared_error(y_test,LR_pred)))
```

```
MAE:  1.4144950285502011
MSE:  3.432437822571741
RMSE:  1.8526839510752342
```

In [7]:

```
#Testing Accuracy
linreg.score(X_test,y_test)
```

Out[7]:

```
0.6703593960062192
```

In [ ]:

```

```

Multiple Linear Regression

In [8]:

```
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_
```

In [9]:

```
from sklearn.linear_model import LinearRegression

multi_lr=LinearRegression()
multi_lr.fit(X_train,y_train)
```

Out[9]:

```
LinearRegression()
```

In [10]:

```
#Prediction
MLR_pred=multi_lr.predict(X_test)
```

In [11]:

```
print("MAE: ",metrics.mean_absolute_error(y_test,MLR_pred))
print("MSE: ",metrics.mean_squared_error(y_test,MLR_pred))
print("RMSE: ",np.sqrt(metrics.mean_squared_error(y_test,MLR_pred)))
```

```
MAE:  1.415219405761792
MSE:  3.4362508969190917
RMSE:  1.8537127331167285
```

In [12]:

```
#Training Accuracy
multi_lr.score(X_train,y_train)
```

Out[12]:

```
0.6717876335584434
```

In [ ]:

```

```

Decision Tree Regression Model

In [13]:

```
# Fitting the Regression model to the dataset
from sklearn.tree import DecisionTreeRegressor
X_train, X_test, y_train, y_test = train_test_split(X, Y) #, test_size=0.3, random_s
regressor = DecisionTreeRegressor(random_state = 0)
regressor.fit(X_train, y_train)
```

Out[13]:

```
DecisionTreeRegressor(random_state=0)
```

In [14]:

```
# Predicting a new result with the Decision Tree Regression
DT_pred = regressor.predict(X_test)
```

In [15]:

```
print("MAE: ",metrics.mean_absolute_error(y_test,DT_pred))
print("MSE: ",metrics.mean_squared_error(y_test,DT_pred))
print("RMSE: ",np.sqrt(metrics.mean_squared_error(y_test,DT_pred)))
```

```
MAE:  0.03433049411174872
MSE:  0.007995524820028228
RMSE:  0.0894176985838275
```

In [16]:

```
regressor.score(X_test,y_test)
```

Out[16]:

```
0.9992337356389224
```

In [ ]:

```

```

Random Forest Regression

In [17]:

```
from sklearn.ensemble import RandomForestRegressor
x_train, x_test, y_train, y_test = train_test_split(X, Y) #, test_size=0.30, random_

RFM = RandomForestRegressor()
RFM.fit(x_train, y_train)
```

Out[17]:

```
RandomForestRegressor()
```

In [18]:

```
RMR_pred = RFM.predict(x_test)
```

In [19]:

```
print("MAE: ",metrics.mean_absolute_error(y_test,RMR_pred))
print("MSE: ",metrics.mean_squared_error(y_test,RMR_pred))
print("RMSE: ",np.sqrt(metrics.mean_squared_error(y_test,RMR_pred)))
```

```
MAE:  0.027316343235186887
MSE:  0.0033653010792856674
RMSE:  0.05801121511643825
```

In [20]:

```
RFM.score(x_test, y_test)
```

Out[20]:

```
0.9996762060735747
```

Support Vector Regression - Warning !!

In [ ]:

```
from sklearn.svm import SVR
x_train, x_test, y_train, y_test = train_test_split(X, Y) #, test_size=0.30, random_

SVR = SVR()
SVR.fit(x_train, y_train)
```

In [ ]:

```
SVR_pred = RFM.predict(x_test)
```

In [ ]:

```
print("MAE: ",metrics.mean_absolute_error(y_test,SVR_pred))
print("MSE: ",metrics.mean_squared_error(y_test,SVR_pred))
print("RMSE: ",np.sqrt(metrics.mean_squared_error(y_test,SVR_pred)))
```

In [ ]:

```
SVR.score(x_test, y_test)
```

In [ ]:

```

```

Logistic Regression

In [ ]:

```
from sklearn.linear_model import LogisticRegression
x_train, x_test, y_train, y_test = train_test_split(X, Y) #, test_size=0.30, random_
```

In [ ]:

```
LogisticModel = LogisticRegression( random_state=0) # solver='liblinear',
```

In [ ]:

```
LogisticModel.fit(x_train, y_train)
```

In [ ]:

```
Logi_pred = LogisticModel.predict(x_test)
```

In [ ]:

```
print("MAE: ",metrics.mean_absolute_error(y_test,Logi_pred))
print("MSE: ",metrics.mean_squared_error(y_test,Logi_pred))
print("RMSE: ",np.sqrt(metrics.mean_squared_error(y_test,Logi_pred)))
```

In [ ]:

```
LogisticModel.score(x_test, y_test)
```

In [ ]:

```

```

Creating Submission File

In [ ]:

```
# CHOOSE YOUR ALGTHM PREDICT LINE AND PLACE  X_test_tasks INSIDE ()

final_pred_test =RFM.predict(X_test_task3)
```

In [ ]:

```
pd.DataFrame({"Id": np.arange(len(final_pred_test)),
              "Category": final_pred_test}).astype(int).to_csv( "2048015_task3_Rando
```

Random Forest Regression and Decision Tree Regression Model having the lowest MSE and RMSE score.

Hence, selecting Random Forest Regression Model for our shared data.