GET A ROOM : ML HACKATHON

Identify the habitability score of a property

Problem Statement

Finding the correct property to live in is a crucial task while moving to a new city/location. An inappropriate property can make our life miserable. Can AI help us find better places?

Task

You have given a relevant dataset about various properties in the USA. Your task is to identify the habitability score of the property.

```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
```

Warning: you are connected to a GPU runtime, but not utilizing the GPU. Change to a standard runtime

```
1 print(train.shape)
2 train.head(2)
```

```
(39499, 15)
```

	Property_ID	Property_Type	Property_Area	Number_of_Windows	Number_of_Doors	Fu
0	0x21e3	Apartment	106	NaN	1	Semi_
1	0x68d4	Apartment	733	2.0	2	Ur

```
1 train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 39499 entries, 0 to 39498
Data columns (total 15 columns):
```

```
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   Property_ID             39499 non-null  object 
 1   Property_Type           39499 non-null  object 
 2   Property_Area           39499 non-null  int64  
 3   Number_of_Windows       37845 non-null  float64
 4   Number_of_Doors         39499 non-null  int64  
 5   Furnishing              38457 non-null  object 
 6   Frequency_of_Powercuts  38116 non-null  float64
 7   Power_Backup            39499 non-null  object 
 8   Water_Supply            39499 non-null  object 
 9   Traffic_Density_Score   39499 non-null  float64
 10  Crime_Rate              38712 non-null  object 
 11  Dust_and_Noise          38280 non-null  object 
 12  Air_Quality_Index       39499 non-null  float64
 13  Neighborhood_Review     39499 non-null  float64
 14  Habitability_score      39499 non-null  float64
dtypes: float64(6), int64(2), object(7)
memory usage: 4.5+ MB
```

```
1 test = pd.read_csv("/content/test.csv")
```

```
1 test.shape
```

```
(10500, 14)
```

Warning: you are connected to a GPU runtime, but not utilizing the GPU. Change to a standard runtime

```
Property_ID                0
Property_Type              0
Property_Area              0
Number_of_Windows       1654
Number_of_Doors            0
Furnishing              1042
Frequency_of_Powercuts  1383
Power_Backup               0
Water_Supply               0
Traffic_Density_Score      0
Crime_Rate               787
Dust_and_Noise          1219
Air_Quality_Index          0
Neighborhood_Review        0
Habitability_score         0
dtype: int64
Property_ID                0
Property_Type              0
Property_Area              0
Number_of_Windows        445
Number_of_Doors            0
Furnishing               257
Frequency_of_Powercuts   366
Power_Backup               0
```

```
Water_Supply                0
Traffic_Density_Score       0
Crime_Rate                212
Dust_and_Noise            330
Air_Quality_Index           0
Neighborhood_Review         0
dtype: int64
```

```
from sklearn.impute import SimpleImputer, KNNImputer
```

```
impute = SimpleImputer(strategy = "most_frequent")
knnimputer = KNNImputer(n_neighbors=10)
```

```
y = train.iloc[:,-1]
train.drop(train.columns[-1],axis=1,inplace=True)
```

```
train2 = train.iloc[:,[3,6]]
train2 = knnimputer.fit_transform(train2)
train2 = pd.DataFrame(train2)
```

```
train3 = train.iloc[:,[5,10,11]]
train3 = impute.fit_transform(train3)
train3 = pd.DataFrame(train3)
```



```
train.iloc[:,3] = train2.iloc[:,0]
train.iloc[:,6] = train2.iloc[:,1]
train.iloc[:,5] = train3.iloc[:,0]
train.iloc[:,10] = train3.iloc[:,1]
train.iloc[:,11] = train3.iloc[:,2]
```

```
test2 = test.iloc[:,[3,6]]
test2 = knnimputer.fit_transform(test2)
test2 = pd.DataFrame(test2)
```

```
test3 = test.iloc[:,[5,10,11]]
test3 = impute.fit_transform(test3)
test3 = pd.DataFrame(test3)
```

```
test.iloc[:,3] = test2.iloc[:,0]
test.iloc[:,6] = test2.iloc[:,1]
test.iloc[:,5] = test3.iloc[:,0]
test.iloc[:,10] = test3.iloc[:,1]
test.iloc[:,11] = test3.iloc[:,2]
```

```
from sklearn.preprocessing import OneHotEncoder
```

```
from sklearn.preprocessing import StandardScaler
ohe = OneHotEncoder()
sc = StandardScaler()
```

```
train2 = train.iloc[:,[1,5,7,8,10,11]]
train2 = ohe.fit_transform(train2)
train2 = pd.DataFrame(train2.toarray())

train3 = train.iloc[:,[2,3,4,6,9,12,13]]
train3 = sc.fit_transform(train3)
train3 = pd.DataFrame(train3)

train = pd.concat([train2,train3],axis=1)
```

```
test2 = test.iloc[:,[1,5,7,8,10,11]]
test2 = ohe.fit_transform(test2)
test2 = pd.DataFrame(test2.toarray())

test3 = test.iloc[:,[2,3,4,6,9,12,13]]
test3 = sc.fit_transform(test3)
test3 = pd.DataFrame(test3)

test = pd.concat([test2,test3],axis=1)
```



```
  from sklearn.tree import DecisionTreeRegressor
  from sklearn.ensemble import RandomForestRegressor
  from sklearn.svm import SVR

  lr = LinearRegression()
  lr.fit(x_train,y_train)
  y_pred = lr.predict(x_test)

  dtr = DecisionTreeRegressor()
  dtr.fit(x_train,y_train)
  y_pred2 = dtr.predict(x_test)

  rfr = RandomForestRegressor()
  rfr.fit(x_train,y_train)
  y_pred3 = rfr.predict(x_test)

  svr = SVR()
  svr.fit(x_train,y_train)
  y_pred4 = svr.predict(x_test)

  return y_pred,y_pred2,y_pred3,y_pred4
```

```
def regressor2(x_train,x_test,y_train,y_test):
```

```
  from sklearn.metrics import mean_absolute_error
  from sklearn.metrics import r2_score
  from sklearn.linear_model import LinearRegression
  from sklearn.tree import DecisionTreeRegressor
  from sklearn.ensemble import RandomForestRegressor
  from sklearn.svm import SVR

  lr = LinearRegression()
  lr.fit(x_train,y_train)
  y_pred = lr.predict(x_test)
  lre = mean_absolute_error(y_test,y_pred)
  print("Linear Regression\nmean absolute error : ",lre,"\n")
  print("R2_score : ",r2_score(y_test,y_pred),"\n")

  dtr = DecisionTreeRegressor()
  dtr.fit(x_train,y_train)
  y_pred2 = dtr.predict(x_test)
  dtre = mean_absolute_error(y_test,y_pred2)
  print("Decision Tree Classifier\nmean absolute error : ",dtre,"\n")
  print("R2_score : ",r2_score(y_test,y_pred2),"\n")

  rfr = RandomForestRegressor()
  rfr.fit(x_train,y_train)
  y_pred3 = rfr.predict(x_test)
  rfre = mean_absolute_error(y_test,y_pred3)
  print("Random Forest Regressor\nmean absolute error : ",rfre,"\n")
```

Warning: you are connected to a GPU runtime, but not utilizing the GPU. Change to a standard runtime

```
  svr = SVR()
  svr.fit(x_train,y_train)
  y_pred4 = svr.predict(x_test)
  svre = mean_absolute_error(y_test,y_pred4)
  print("Suppoer Vector Regression\nmean absolute error : ",svre,"\n")
  print("R2_score : ",r2_score(y_test,y_pred4),"\n")

  return y_pred,y_pred2,y_pred3,y_pred4
```

```
p1,p2,p3,p4 = regressor1(train,test,y)
```

```
from sklearn.metrics import mean_absolute_error
mean_absolute_error(p1,p2)
```

```
7.393517008928571
```

```
mean_absolute_error(p2,p3)
```

```
4.156624790476189
```

```
mean_absolute_error(p3,p4)
```

```
    3.58908326486187
```

```
1 mean_absolute_error(p4,p2)
```

```
    5.812582066087356
```

```
1 p3
```

```
    array([30.5028, 80.1084, 67.392 , ..., 75.1799, 79.6257, 79.4099])
```

```
1 test = pd.read_csv("/content/test.csv")
2 id = test.iloc[:,0]
```

```
1 submission = pd.DataFrame({"Property_ID":id,"Habitability_score":p3})
```

```
1 submission
```

	Property_ID	Habitability_score
0	0x6e93	30.5028
1	0x8787	80.1084
3	0x9dbd	73.0809
4	0xbfde	75.4409
...	...	...
10495	0x423d	64.4912
10496	0x78c5	81.0152
10497	0xbf3	75.1799
10498	0x305b	79.6257
10499	0x5cff	79.4099

10500 rows × 2 columns


```
1 submission.to_csv("Submission.csv",index=False)
```

```
1 pd.read_csv("Submission.csv")
```

	Property_ID	Habitability_score
0	0x6e93	30.5028
1	0x8787	80.1084
2	0x6c17	67.3920
3	0x9dbd	73.0809
4	0xbfde	75.4409
...	...	...
10495	0x423d	64.4912
10496	0x78c5	81.0152
10497	0xbf3	75.1799
10498	0x305b	79.6257
10499	0x5cff	79.4099

```
1
```

Warning: you are connected to a GPU runtime, but not utilizing the GPU. Change to a standard runtime

✓ 0s completed at 9:30 AM