Parts A, B, C and D: Create Data by sampling from Gaussians, create dependent variables, add noise to feature inputs

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.mixture import GaussianMixture
from scipy.stats import norm

feature_1 = np.random.normal(0, 0.5, size=200)
feature_2 = np.random.normal(1, 0.2, size=200)
feature_3 = np.random.normal(-1, 0.1, size=200)

X=[]
Y=[]

for i in range(len(feature_1)):
  X.append([feature_1[i], feature_2[i], feature_3[i]])
  Y.append(feature_1[i] + 2*feature_2[i] + 3*feature_3[i])

noisy_feature_1 = feature_1 + np.random.normal(0, 0.1, 200)
noisy_feature_2 = feature_1 + np.random.normal(0, 0.1, 200)
noisy_feature_3 = feature_1 + np.random.normal(0, 0.1, 200)

noisy_X=[]

for i in range(len(feature_1)):
  noisy_X.append([noisy_feature_1[i], noisy_feature_2[i], noisy_feature_3[i]])
```

Part E: Employ MLR and calculate R^2 score on samples @70-30 split

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from tqdm import tqdm
import sklearn.model_selection
```

```
from sklearn.utils import resample

bootstrapped_r_2_scores_train=[]
bootstrapped_r_2_scores_test=[]
n_bootstrap=1000

for x in tqdm(list(range(n_bootstrap))):
  X_train, X_test, y_train, y_test = train_test_split(noisy_X, Y, test_size=0.3, random_state=np.random.randint(0, 50))
  reg=LinearRegression().fit(X_train, y_train)

  y_pred_train = reg.predict(X_train)
  y_pred_test = reg.predict(X_test)

  train_r = r2_score(y_train, y_pred_train)
  test_r = r2_score(y_test, y_pred_test)

  bootstrapped_r_2_scores_train.append(train_r)
  bootstrapped_r_2_scores_test.append(test_r)

print(bootstrapped_r_2_scores_test)
print(bootstrapped_r_2_scores_train)
```

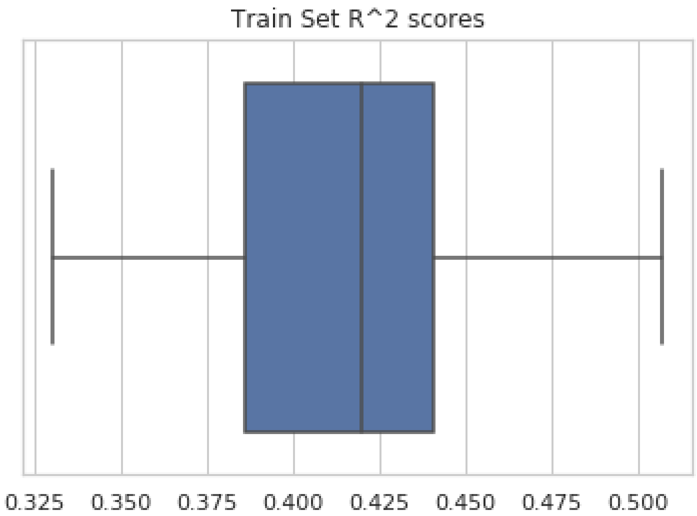
```
100%|██████████| 1000/1000 [00:01<00:00, 541.90it/s][0.44700982100253206, 0.4919574291941685, 0.2483378626588567, 0.2
[0.3790346305856289, 0.3442686560028939, 0.4344761970094305, 0.4360884428214884, 0.4077289820934752, 0.42142864412024
```

Part F: Plot R^2 scores as a boxplot

```
import seaborn as sns
sns.set(style="whitegrid")
ax = sns.boxplot(x=bootstrapped_r_2_scores_test)
plt.title('Test Set R^2 scores')
plt.show()

ax = sns.boxplot(x=bootstrapped_r_2_scores_train)
plt.title('Train Set R^2 scores')
```

```
plt.show()
```



Explanation:

I have sampled the three different feature sets using the following gaussian distributions for each set:

- Feature 1: ~N(0, 0.5)

- Feature 2: ~N(1, 0.2)
- Feature 3: ~N(-1, 0.1)

The dependent variable Y has been created in the following manner: Y = X1 + 2*X2 + 3*X3

To create the noisy inputs, I have added a random noise sampled from ~N(0, 0.1) to each feature input.

For the bootstrapping of samples, in each iteration (1000 iterations in total), I have considered the 200 samples to be my sample set. I then randomly sample out 70% points as the train set and the remaining 30% random points as the test set. I then apply MLR using the following formula:

Y = B0 + B1*X1 + B2*X2 + B3*X3 + E, where B0 is the constant bias term, Bi are the coefficients for the individual features, and E is a random noise term.

Once, I fit a MLR model on each bootstrapped train test split sample, I measure the R^2 score for the predictions made by the MLR model. The distribution of the R^2 scores over the 1000 bootstrapped samples are shown in the plots above (for both train and test sets).