Task 1

In [1]:
```
#Taking housing prices dataset from kaggle to work on this task
```

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

In [3]:
```
df=pd.read_csv("./Housing_train.csv")
```

In [4]:
```
df.head()
```

Out[4]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl

5 rows × 81 columns

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1460 entries, 0 to 1459
Data columns (total 81 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             1460 non-null   int64  
 1   MSSubClass     1460 non-null   int64  
 2   MSZoning       1460 non-null   object 
 3   LotFrontage    1201 non-null   float64
 4   LotArea        1460 non-null   int64  
 5   Street         1460 non-null   object 
 6   Alley          91 non-null     object 
 7   LotShape       1460 non-null   object 
 8   LandContour    1460 non-null   object 
 9   Utilities      1460 non-null   object 
 10  LotConfig      1460 non-null   object 
 11  LandSlope      1460 non-null   object 
 12  Neighborhood   1460 non-null   object 
 13  Condition1     1460 non-null   object 
 14  Condition2     1460 non-null   object 
 15  BldgType       1460 non-null   object 
 16  HouseStyle     1460 non-null   object 
 17  OverallQual    1460 non-null   int64  
 18  OverallCond    1460 non-null   int64  
 19  YearBuilt      1460 non-null   int64  
 20  YearRemodAdd   1460 non-null   int64  
 21  RoofStyle      1460 non-null   object 
 22  RoofMatl       1460 non-null   object 
 23  Exterior1st    1460 non-null   object 
 24  Exterior2nd    1460 non-null   object 
 25  MasVnrType     1452 non-null   object 
 26  MasVnrArea     1452 non-null   float64
 27  ExterQual      1460 non-null   object 
 28  ExterCond      1460 non-null   object 
 29  Foundation     1460 non-null   object 
 30  BsmtQual       1423 non-null   object 
 31  BsmtCond       1423 non-null   object 
 32  BsmtExposure   1422 non-null   object 
 33  BsmtFinType1   1423 non-null   object 
 34  BsmtFinSF1     1460 non-null   int64  
 35  BsmtFinType2   1422 non-null   object 
 36  BsmtFinSF2     1460 non-null   int64  
 37  BsmtUnfSF      1460 non-null   int64  
 38  TotalBsmtSF    1460 non-null   int64  
 39  Heating        1460 non-null   object 
 40  HeatingQC      1460 non-null   object 
 41  CentralAir     1460 non-null   object 
 42  Electrical     1459 non-null   object 
 43  1stFlrSF       1460 non-null   int64  
 44  2ndFlrSF       1460 non-null   int64  
 45  LowQualFinSF   1460 non-null   int64  
 46  GrLivArea      1460 non-null   int64  
 47  BsmtFullBath   1460 non-null   int64  
 48  BsmtHalfBath   1460 non-null   int64  
 49  FullBath       1460 non-null   int64  
 50  HalfBath       1460 non-null   int64  
 51  BedroomAbvGr   1460 non-null   int64  
```

```
 52  KitchenAbvGr   1460 non-null   int64
 53  KitchenQual    1460 non-null   object
 54  TotRmsAbvGrd   1460 non-null   int64
 55  Functional     1460 non-null   object
 56  Fireplaces     1460 non-null   int64
 57  FireplaceQu    770 non-null    object
 58  GarageType     1379 non-null   object
 59  GarageYrBlt    1379 non-null   float64
 60  GarageFinish   1379 non-null   object
 61  GarageCars     1460 non-null   int64
 62  GarageArea     1460 non-null   int64
 63  GarageQual     1379 non-null   object
 64  GarageCond     1379 non-null   object
 65  PavedDrive     1460 non-null   object
 66  WoodDeckSF     1460 non-null   int64
 67  OpenPorchSF    1460 non-null   int64
 68  EnclosedPorch  1460 non-null   int64
 69  3SsnPorch      1460 non-null   int64
 70  ScreenPorch    1460 non-null   int64
 71  PoolArea       1460 non-null   int64
 72  PoolQC         7 non-null      object
 73  Fence          281 non-null    object
 74  MiscFeature    54 non-null     object
 75  MiscVal        1460 non-null   int64
 76  MoSold         1460 non-null   int64
 77  YrSold         1460 non-null   int64
 78  SaleType       1460 non-null   object
 79  SaleCondition  1460 non-null   object
 80  SalePrice      1460 non-null   int64
dtypes: float64(3), int64(35), object(43)
memory usage: 924.0+ KB
```

In [6]:

```
#checking for null values
print(df.isnull().sum())
```

```
Id                 0
MSSubClass         0
MSZoning           0
LotFrontage      259
LotArea            0
                ... 
MoSold             0
YrSold             0
SaleType           0
SaleCondition      0
SalePrice          0
Length: 81, dtype: int64
```

In [7]:

```
#filling the null values with most common values
df.fillna(df.mode().iloc[0], inplace=True)
```

In [8]:
```
#standardising the values
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
```

In [9]:
```
#numerical values
df.select_dtypes(include=np.number).columns.tolist()
```

Out[9]:
```
['Id',
 'MSSubClass',
 'LotFrontage',
 'LotArea',
 'OverallQual',
 'OverallCond',
 'YearBuilt',
 'YearRemodAdd',
 'MasVnrArea',
 'BsmtFinSF1',
 'BsmtFinSF2',
 'BsmtUnfSF',
 'TotalBsmtSF',
 '1stFlrSF',
 '2ndFlrSF',
 'LowQualFinSF',
 'GrLivArea',
 'BsmtFullBath',
 'BsmtHalfBath',
 'FullBath',
 'HalfBath',
 'BedroomAbvGr',
 'KitchenAbvGr',
 'TotRmsAbvGrd',
 'Fireplaces',
 'GarageYrBlt',
 'GarageCars',
 'GarageArea',
 'WoodDeckSF',
 'OpenPorchSF',
 'EnclosedPorch',
 '3SsnPorch',
 'ScreenPorch',
 'PoolArea',
 'MiscVal',
 'MoSold',
 'YrSold',
 'SalePrice']
```

In [10]:
```
num=df.select_dtypes(include=np.number).columns.tolist()
df[num]=scaler.fit_transform(df[num])
```

In [11]:

```
df[num]
```

Out[11]:

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt
0	-1.730865	0.073375	-0.146189	-0.207142	0.651479	-0.517200	1.050994
1	-1.728492	-0.872563	0.524992	-0.091886	-0.071836	2.179628	0.156734
2	-1.726120	0.073375	-0.011953	0.073480	0.651479	-0.517200	0.984752
3	-1.723747	0.309859	-0.369915	-0.096897	0.651479	-0.517200	-1.863632
4	-1.721374	0.073375	0.703973	0.375148	1.374795	-0.517200	0.951632
...	...	...	...	...	...	...	...
1455	1.721374	0.073375	-0.280425	-0.260560	-0.071836	-0.517200	0.918511
1456	1.723747	-0.872563	0.748718	0.266407	-0.071836	0.381743	0.222975
1457	1.726120	0.309859	-0.101443	-0.147810	0.651479	3.078570	-1.002492
1458	1.728492	-0.872563	-0.011953	-0.080160	-0.795151	0.381743	-0.704406
1459	1.730865	-0.872563	0.301265	-0.058112	-0.795151	0.381743	-0.207594

1460 rows × 38 columns

In [12]:

```
#handling categorical values
from sklearn.preprocessing import LabelEncoder
categorical=df.select_dtypes(include=[object]).columns.tolist()
le=LabelEncoder()
df[categorical] = df[categorical].apply(lambda col: le.fit_transform(col))
df.head()
```

Out[12]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	Lan
0	-1.730865	0.073375	3	-0.146189	-0.207142	1	0	3	
1	-1.728492	-0.872563	3	0.524992	-0.091886	1	0	3	
2	-1.726120	0.073375	3	-0.011953	0.073480	1	0	0	
3	-1.723747	0.309859	3	-0.369915	-0.096897	1	0	0	
4	-1.721374	0.073375	3	0.703973	0.375148	1	0	0	

5 rows × 81 columns

In [13]:
```
#using one hot encoding
onehotencoding=pd.get_dummies(df, drop_first=True)
onehotencoding.head()
```

Out[13]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	Lan
0	-1.730865	0.073375	3	-0.146189	-0.207142	1	0	3	
1	-1.728492	-0.872563	3	0.524992	-0.091886	1	0	3	
2	-1.726120	0.073375	3	-0.011953	0.073480	1	0	0	
3	-1.723747	0.309859	3	-0.369915	-0.096897	1	0	0	
4	-1.721374	0.073375	3	0.703973	0.375148	1	0	0	

5 rows × 81 columns

In [14]:
```
#multicollinearity
#the more the value the more collinear the features are
df.corr()
```

Out[14]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Al
Id	1.000000	0.011156	-0.006096	-0.012497	-0.033226	0.008916	-0.001
MSSubClass	0.011156	1.000000	0.035900	-0.349116	-0.139781	-0.024969	0.184
MSZoning	-0.006096	0.035900	1.000000	-0.101150	-0.034452	0.087654	-0.329
LotFrontage	-0.012497	-0.349116	-0.101150	1.000000	0.281283	-0.037078	-0.159
LotArea	-0.033226	-0.139781	-0.034452	0.281283	1.000000	-0.197131	-0.077
...	...	...	...	...	...	...	
MoSold	0.021172	-0.013585	-0.031496	0.012785	0.001205	0.003690	-0.021
YrSold	0.000712	-0.021407	-0.020628	0.003021	-0.014261	-0.025043	-0.001
SaleType	0.019773	0.012464	0.097437	-0.035773	0.012292	0.014339	-0.005
SaleCondition	-0.005806	-0.024940	0.009494	0.061393	0.034169	0.006064	-0.028
SalePrice	-0.021917	-0.084284	-0.166872	0.329220	0.263843	0.041036	-0.027

81 rows × 81 columns

Task 2

In [15]:
```
import pandas as pd
```

In [16]:
```
# Not using only 643 rows becuase the whole data set was crashing
# the system while performming label encoding
```

In [17]:
```
with open('./farm-data/farm-ads.txt') as f:
    lines = f.readlines()
len(lines)
temp=[]
for i in range(len(lines)-3500):
    temp.append(lines[i].split())
farm_ad=pd.DataFrame(temp)
farm_ad.head()
```

Out[17]:

	0	1	2	3	4	5	6	7	
0	1	ad-jerry	ad-bruckheimer	ad-chase	ad-premier	ad-sept	ad-th	ad-clip	bruckheir
1	-1	ad-rheumatoid	ad-arthritis	ad-expert	ad-tip	ad-info	ad-article	ad-treatment	ad-opt
2	-1	ad-rheumatologist	ad-anju	ad-varghese	ad-yonker	ad-ny	ad-pomona	ad-ny	ad-w
3	-1	ad-siemen	ad-water	ad-remediation	ad-water	ad-scarce	ad-resource	ad-siemen	ad-h
4	-1	ad-symptom	ad-muscle	ad-weakness	ad-genetic	ad-disease	ad-symptom	ad-include	ad-sea

5 rows × 7373 columns

In [18]:
```
with open('./farm-data/farm-ads-vect.txt') as f:
    lines = f.readlines()
len(lines)
temp=[]
for i in range(len(lines)-3500):
    temp.append(lines[i].split())
farm_ad_vec=pd.DataFrame(temp)
farm_ad_vec.head()
```

Out[18]:

	0	1	2	3	4	5	6	7	8	9	...	1737	1738	1739	17
0	1	1:1	2:1	3:1	4:1	5:1	6:1	7:1	8:1	9:1	...	None	None	None	Nc
1	-1	10:1	11:1	12:1	13:1	14:1	15:1	16:1	17:1	18:1	...	None	None	None	Nc
2	-1	29:1	31:1	35:1	101:1	131:1	252:1	272:1	280:1	291:1	...	None	None	None	Nc
3	-1	34:1	35:1	36:1	44:1	54:1	84:1	94:1	104:1	126:1	...	None	None	None	Nc
4	-1	8:1	9:1	429:1	430:1	431:1	432:1	433:1	434:1	435:1	...	None	None	None	Nc

5 rows × 1747 columns

In [19]:
```
data=pd.merge(farm_ad,farm_ad_vec,left_index=True,right_index=True)
data
```

Out[19]:

	0_x	1_x	2_x	3_x	4_x	5_x	6_x	7_x
0	1	ad-jerry	ad-bruckheimer	ad-chase	ad-premier	ad-sept	ad-th	ad-clip
1	-1	ad-rheumatoid	ad-arthritis	ad-expert	ad-tip	ad-info	ad-article	ad-treatmen
2	-1	ad-rheumatologist	ad-anju	ad-varghese	ad-yonker	ad-ny	ad-pomona	ad-ny
3	-1	ad-siemen	ad-water	ad-remediation	ad-water	ad-scarce	ad-resource	ad-siemen
4	-1	ad-symptom	ad-muscle	ad-weakness	ad-genetic	ad-disease	ad-symptom	ad-include
...	...	...	...	...	...	...	...	..
638	-1	ad-fibromyalgia	ad-free	ad-hopkin	ad-fibromyalgia	ad-report	ad-learn	ad-stop
639	-1	ad-infectious	ad-disease	ad-online	ad-cost	ad-conference	ad-cme	ad-credi
640	-1	ad-local	ad-sperm	ad-bank	ad-sperm	ad-donor	ad-pregnant	ad-reques
641	1	ad-aoudad	ad-whitetail	ad-hunt	ad-ram	ad-white	ad-tail	ad-low
642	-1	ad-egg	ad-donation	ad-service	ad-low	ad-cost	ad-birth	ad-success

643 rows × 9120 columns

In [20]:
```
from sklearn.preprocessing import LabelEncoder
categorical=data.select_dtypes(include=[object]).columns.tolist()
le=LabelEncoder()
data[categorical] = data[categorical].apply(lambda col: le.fit_transform(c
# data['0_y'] = data['0_y'].apply(lambda col: le.fit_transform(col))
# data['0_x']=data['0_x'].astype('category').cat.codes
# data['0_y']=data['0_y'].astype('category').cat.codes
```

In [21]:

```
# data['0_x']=data['0_x'].astype('category').cat.codes
# data['0_y']=data['0_y'].astype('category').cat.codes
data['target'] = data['0_x'] + data['0_y']
data = data.drop(['0_x','0_y'],axis = 1)
data
```

```
C:\Users\Aaryan Agarwal\AppData\Local\Temp\ipykernel_31852\3533153650.py:
3: PerformanceWarning: DataFrame is highly fragmented.  This is usually t
he result of calling `frame.insert` many times, which has poor performanc
e.  Consider joining all columns at once using pd.concat(axis=1) instead.
To get a de-fragmented frame, use `newframe = frame.copy()`
  data['target'] = data['0_x'] + data['0_y']
```

Out[21]:

	1_x	2_x	3_x	4_x	5_x	6_x	7_x	8_x	9_x	10_x	...	1738_y	1739_y	1740_y	1741
0	162	29	40	218	268	317	53	23	33	212	...	1	1	1	
1	251	10	99	295	156	12	315	182	217	363	...	1	1	1	
2	252	6	303	327	200	226	209	293	12	40	...	1	1	1	
3	263	310	232	317	260	265	277	118	159	350	...	1	1	1	
4	282	184	310	129	89	306	156	228	115	212	...	1	1	1	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
638	112	112	135	120	249	170	290	186	307	302	...	1	1	1	
639	152	80	195	64	61	52	61	135	98	175	...	1	1	1	
640	177	265	14	272	96	232	256	107	106	102	...	1	1	1	
641	17	313	143	237	334	307	189	105	173	152	...	1	1	1	
642	93	82	250	169	65	21	293	212	51	25	...	1	1	1	

643 rows × 9119 columns

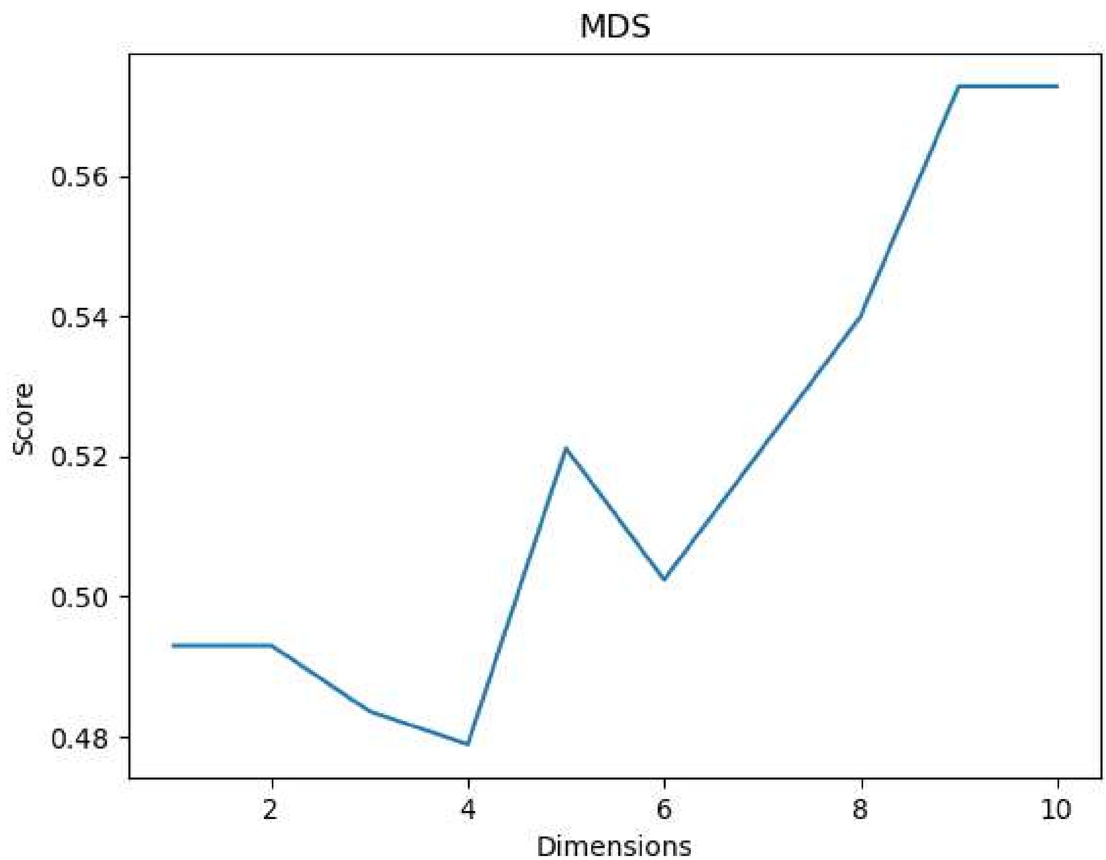
In [22]:

```
#MDS
from scipy.spatial.distance import pdist,squareform
from sklearn.manifold import MDS
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

X = data.drop('target',axis = 1)
y = data['target']
scores=[]
dismat=squareform(pdist(data))

for i in range(1,11):
    mds=MDS(n_components=i,dissimilarity='precomputed')
    dim_red=mds.fit_transform(dismat)
    X_train, X_test, y_train, y_test = train_test_split(dim_red, y, test_s
    lr=LogisticRegression()
    lr.fit(X_train,y_train)
    scores.append(lr.score(X_test,y_test))
print(scores)
```

```
[0.49295774647887325, 0.49295774647887325, 0.4835680751173709, 0.47887323
94366197, 0.5211267605633803, 0.5023474178403756, 0.5211267605633803, 0.5
39906103286385, 0.5727699530516432, 0.5727699530516432]
```

In [23]:

```
import matplotlib.pyplot as plt
plt.title("MDS")
plt.xlabel("Dimensions")
plt.ylabel("Score")
plt.plot(list(range(1,11)), scores)
plt.show()
```

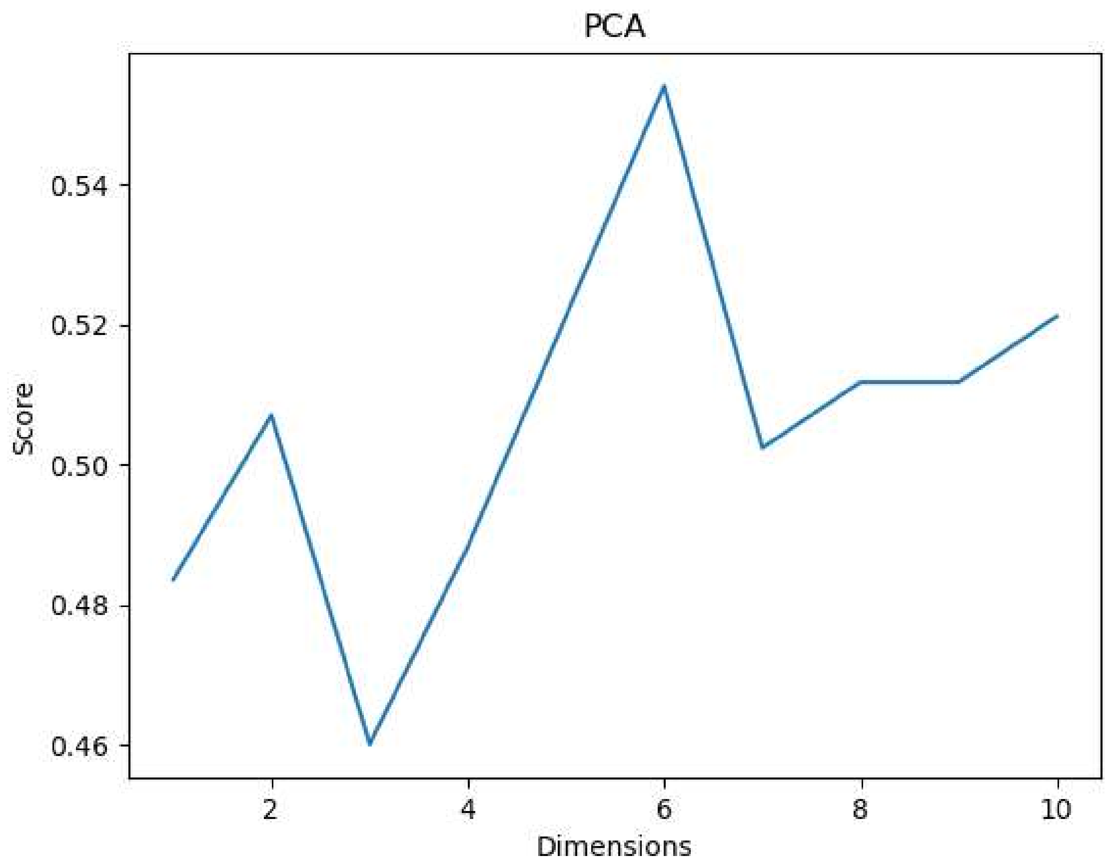


In [24]:

```
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
X = data.drop('target',axis = 1)
y = data['target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33,
scores=[]
for i in range(1,11):
    alg=PCA(n_components=i)
    X_train_reduced = alg.fit_transform(X_train)
    lr = LogisticRegression()
    lr.fit(X_train_reduced, y_train)
    X_test_reduced = alg.transform(X_test)
    scores.append(lr.score(X_test_reduced, y_test))
scores
```

```
C:\Users\Aaryan Agarwal\anaconda3\lib\site-packages\sklearn\utils\vali
dation.py:1688: FutureWarning: Feature names only support names that a
re all strings. Got feature names with dtypes: ['int', 'str']. An erro
r will be raised in 1.2.
  warnings.warn(
C:\Users\Aaryan Agarwal\anaconda3\lib\site-packages\sklearn\utils\vali
dation.py:1688: FutureWarning: Feature names only support names that a
re all strings. Got feature names with dtypes: ['int', 'str']. An erro
r will be raised in 1.2.
  warnings.warn(
C:\Users\Aaryan Agarwal\anaconda3\lib\site-packages\sklearn\utils\vali
dation.py:1688: FutureWarning: Feature names only support names that a
re all strings. Got feature names with dtypes: ['int', 'str']. An erro
r will be raised in 1.2.
  warnings.warn(
C:\Users\Aaryan Agarwal\anaconda3\lib\site-packages\sklearn\utils\vali
dation.py:1688: FutureWarning: Feature names only support names that a
re all strings. Got feature names with dtypes: ['int', 'str']. An erro
r will be raised in 1.2.
```

In [25]:
```
plt.plot(list(range(1,11)), scores)
plt.title("PCA")
plt.xlabel("Dimensions")
plt.ylabel("Score")
plt.show()
```



In [26]:
```
#LDA
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
X = data.drop('target',axis = 1)
y = data['target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33,
scores=[]

# X_lda = lda.fit(X, y).transform(X)
lda = LinearDiscriminantAnalysis(n_components=1)
X_train_reduced = lda.fit(X_train,y_train).transform(X_train)
lr = LogisticRegression()
lr.fit(X_train_reduced, y_train)
X_test_reduced = lda.transform(X_test)
scores.append(lr.score(X_test_reduced, y_test))
scores
```

```
C:\Users\Aaryan Agarwal\anaconda3\lib\site-packages\sklearn\utils\validat
ion.py:1688: FutureWarning: Feature names only support names that are all
strings. Got feature names with dtypes: ['int', 'str']. An error will be
raised in 1.2.
  warnings.warn(
C:\Users\Aaryan Agarwal\anaconda3\lib\site-packages\sklearn\utils\validat
ion.py:1688: FutureWarning: Feature names only support names that are all
strings. Got feature names with dtypes: ['int', 'str']. An error will be
raised in 1.2.
  warnings.warn(
C:\Users\Aaryan Agarwal\anaconda3\lib\site-packages\sklearn\utils\validat
ion.py:1688: FutureWarning: Feature names only support names that are all
strings. Got feature names with dtypes: ['int', 'str']. An error will be
raised in 1.2.
  warnings.warn(
```

Out[26]:
```
[0.568075117370892]
```

Task 3

In [27]:
```
df=pd.read_csv("./BCI-SSVEP_Database_Aceves/A002SB3_1.csv")
```

In [28]:

```
df.head()
```

Out[28]:

	title:C	recorded:01.07.20 12.54.57	sampling:128	subject:C	labels:COUNTER INTERPOLATED AF3 F7 F3 FC5 T7 P7 O1 O2 P8 T8 FC6 F4 F8 AF4	chan:37	unit
0	48	0	4226.666563	4219.487076	4215.897333	4215.897333	421
1	49	0	4227.692204	4214.871692	4215.384512	4217.435794	422
2	50	0	4230.769127	4224.102461	4216.410153	4223.076820	422
3	51	0	4231.281948	4228.717845	4221.538358	4225.128102	421
4	52	0	4226.666563	4219.999897	4215.897333	4218.461435	421

In [29]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9600 entries, 0 to 9599
Data columns (total 16 columns):
 #   Column
Non-Null Count  Dtype
---  ------
--------------  -----
 0   title:C
9600 non-null   int64
 1    recorded:01.07.20 12.54.57
9600 non-null   int64
 2    sampling:128
9600 non-null   float64
 3    subject:C
9600 non-null   float64
 4    labels:COUNTER INTERPOLATED AF3 F7 F3 FC5 T7 P7 O1 O2 P8 T8 FC6 F4
F8 AF4                  9600 non-null   float64
 5    chan:37
9600 non-null   float64
 6    units:emotiv
9600 non-null   float64
 7   Unnamed: 7
9600 non-null   float64
 8   Unnamed: 8
9600 non-null   float64
 9   Unnamed: 9
9600 non-null   float64
 10  Unnamed: 10
9600 non-null   float64
 11  Unnamed: 11
9600 non-null   float64
 12  Unnamed: 12
9600 non-null   float64
 13  Unnamed: 13
9600 non-null   float64
 14  Unnamed: 14
9600 non-null   float64
 15  Unnamed: 15
9600 non-null   float64
dtypes: float64(14), int64(2)
memory usage: 1.2 MB
```

In [30]:
```
df.corr()
```

Out[30]:

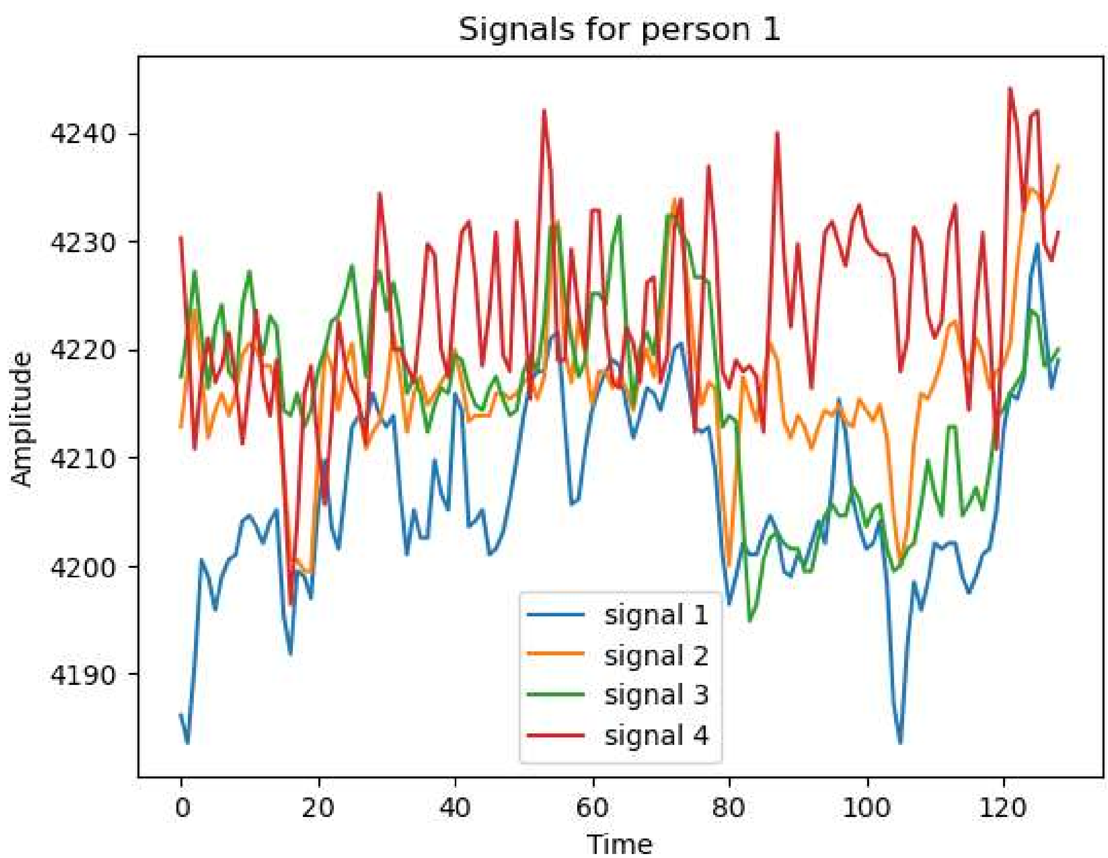
	title:C	recorded:01.07.20 12.54.57	sampling:128	subject:C	labels:COUNTER INTERPOLATED AF3 F7 F3 FC5 T7 P7 O1 O2 P8 T8 FC6 F4 F8 AF4
title:C	1.000000	NaN	0.033072	-0.013029	0.019828
recorded:01.07.20 12.54.57	NaN	NaN	NaN	NaN	NaN
sampling:128	0.033072	NaN	1.000000	0.097258	-0.000708
subject:C	-0.013029	NaN	0.097258	1.000000	0.183393
labels:COUNTER INTERPOLATED AF3 F7 F3 FC5 T7 P7 O1 O2 P8 T8 FC6 F4 F8 AF4	0.019828	NaN	-0.000708	0.183393	1.000000
chan:37	-0.046208	NaN	-0.339962	0.110888	0.320856
units:emotiv	0.023007	NaN	-0.010816	0.045856	0.161736
Unnamed: 7	-0.030816	NaN	-0.076676	0.072644	0.275128
Unnamed: 8	-0.033949	NaN	0.047691	0.032575	0.233974
Unnamed: 9	-0.033426	NaN	0.013321	0.090028	0.004276
Unnamed: 10	-0.038339	NaN	0.044370	0.093762	0.254246
Unnamed: 11	-0.030735	NaN	-0.090107	0.112440	0.058015
Unnamed: 12	0.029623	NaN	-0.095490	-0.060605	0.428704
Unnamed: 13	0.028798	NaN	0.074780	0.051832	0.480984
Unnamed: 14	0.046023	NaN	0.002888	-0.004076	0.378384
Unnamed: 15	-0.012715	NaN	0.206420	0.141856	0.183190

In [31]:
```
import matplotlib.pyplot as plt
```

In [32]:
```
x=df.iloc[81:210,0]
y=[df.iloc[81:210,3],df.iloc[81:210,4],df.iloc[81:210,5],df.iloc[81:210,6]
```

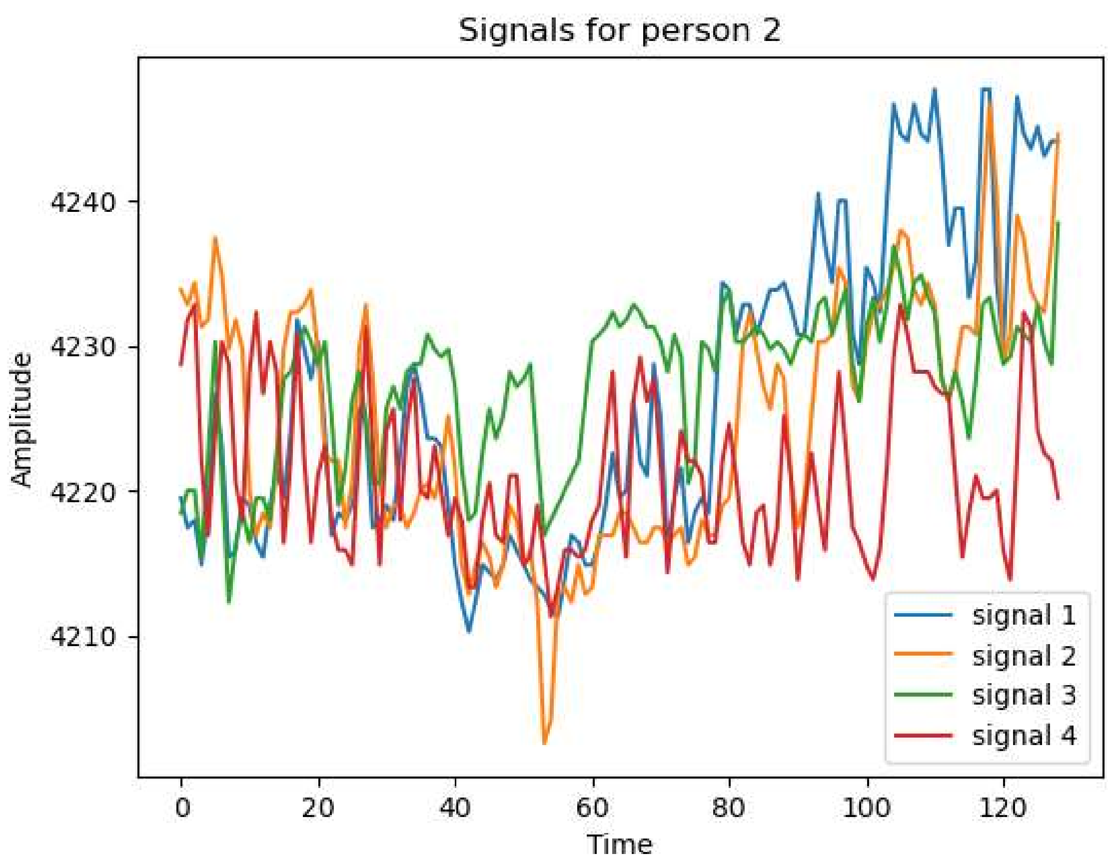
In [33]:

```
plt.plot(x,y[0],label="signal 1")
plt.plot(x,y[1],label="signal 2")
plt.plot(x,y[2],label="signal 3")
plt.plot(x,y[3],label="signal 4")

plt.title("Signals for person 1")
plt.xlabel("Time")
plt.ylabel("Amplitude")
plt.legend()
plt.show()
# the graph shows the brain signal for person 1
```

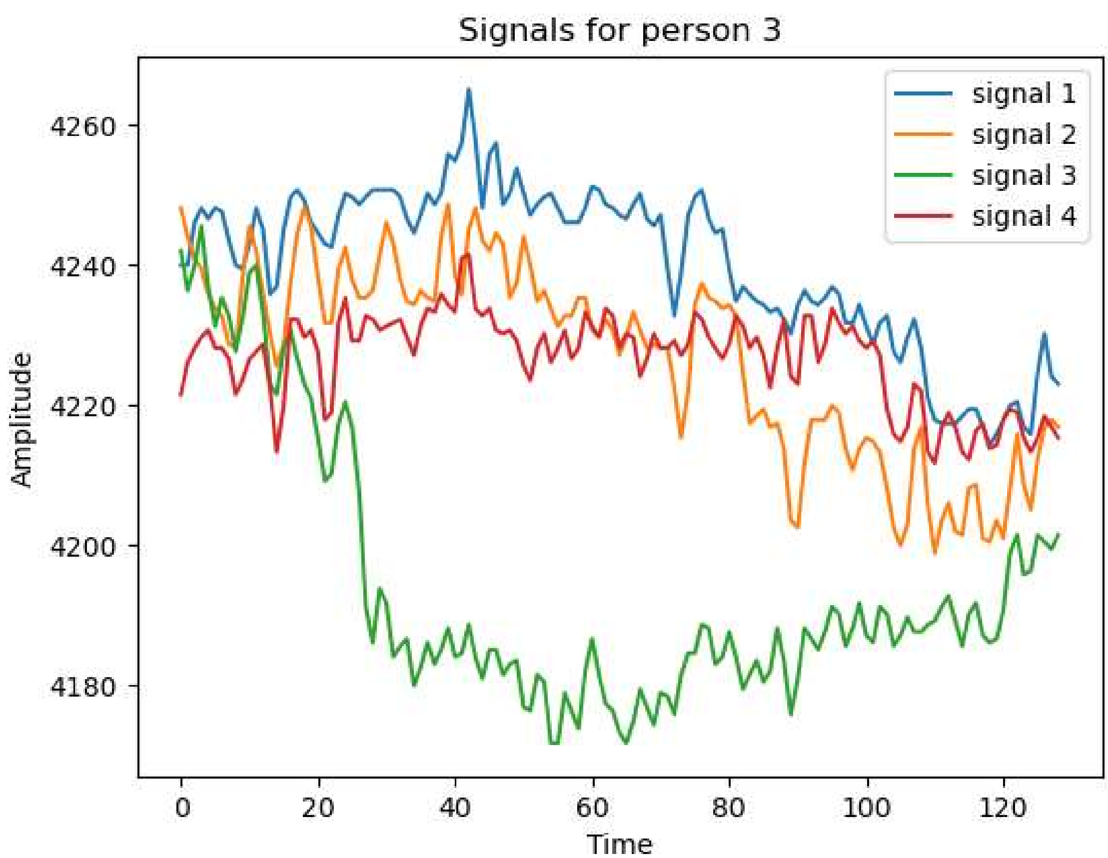


In [34]:

```
x=df.iloc[210:339,0]
y=[df.iloc[210:339,3],df.iloc[210:339,4],df.iloc[210:339,5],df.iloc[210:33
plt.plot(x,y[0],label="signal 1")
plt.plot(x,y[1],label="signal 2")
plt.plot(x,y[2],label="signal 3")
plt.plot(x,y[3],label="signal 4")

plt.title("Signals for person 2")
plt.xlabel("Time")
plt.ylabel("Amplitude")
plt.legend()
plt.show()
# the graph shows the brain signals from person 2
```



In [35]:
```
x=df.iloc[339:468,0]
y=[df.iloc[339:468,3],df.iloc[339:468,4],df.iloc[339:468,5],df.iloc[339:46
plt.plot(x,y[0],label="signal 1")
plt.plot(x,y[1],label="signal 2")
plt.plot(x,y[2],label="signal 3")
plt.plot(x,y[3],label="signal 4")

plt.title("Signals for person 3")
plt.xlabel("Time")
plt.ylabel("Amplitude")
plt.legend()
plt.show()
# the graph shows the brain signals for person 3
```



In [36]:
```
df=pd.read_csv("./repeat_consumption_data/data/reddit_sample/test.csv",hea
```

In [37]:
```
df.head()
```

Out[37]:

	0	1	2
0	1	575	4
1	1	3063	2
2	1	5576	3
3	1	11561	1
4	1	11835	2

In [38]:
```
df.corr()
```

Out[38]:

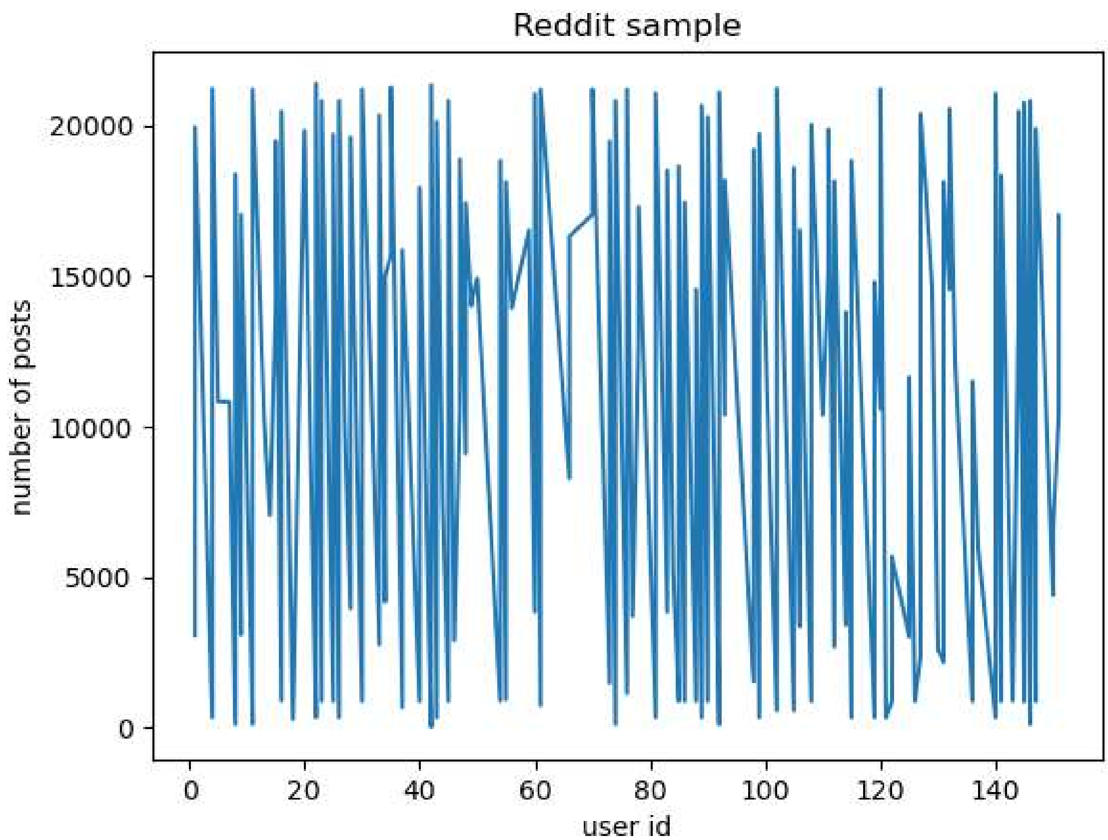
	0	1	2
0	1.000000	0.002036	-0.011609
1	0.002036	1.000000	-0.015056
2	-0.011609	-0.015056	1.000000

In [39]:
```
# for i in range(10):
signal = df.iloc[1:1000]
print(signal)
```

```
       0      1  2
1      1   3063  2
2      1   5576  3
3      1  11561  1
4      1  11835  2
5      1  12396  1
..   ...    ... ..
995  150   5947  1
996  150   5958  1
997  150   6569  6
998  151  10361  1
999  151  17016  3

[999 rows x 3 columns]
```

In [40]:
```
x=signal[0]
y=signal[1]
```

In [41]:
```
plt.plot(x,y)
plt.title("Reddit sample")
plt.xlabel("user id")
plt.ylabel("number of posts")
plt.show()
```



In [ ]:
```
# the number of posts made by each user id
```

In [ ]:
```
```