Problem1

In [82]:
```
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import scipy
from sklearn.impute import SimpleImputer
from sklearn.decomposition import PCA
from scipy import spatial
from numpy.linalg import norm
import math
from sklearn.metrics import jaccard_score
```

In [2]:
```
data=sns.load_dataset('titanic')
```

In [3]:
```
data.head()
```

Out[3]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True

In [4]:
```
data.shape
```

Out[4]:
```
(891, 15)
```

In [5]:
```
data['survived'].value_counts()
```

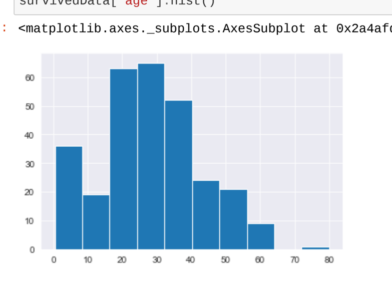
Out[5]:
```
0    549
1    342
Name: survived, dtype: int64
```

In [6]:
```
survivedData=data[data["survived"]==1]
survivedData.shape
```

Out[6]:
```
(342, 15)
```

In [7]:
```
sns.set_style('darkgrid')
survivedData['age'].hist()
```

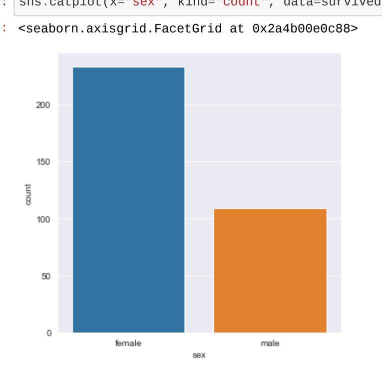
Out[7]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2a4afdc3808>
```



Of the 342 people survived maimum number of the survivors are from the age group of 20 to 40 followed by age group of 0 to 10

In [8]:
```
sns.catplot(x="sex", kind="count", data=survivedData)
```

Out[8]:
```
<seaborn.axisgrid.FacetGrid at 0x2a4b00e0c88>
```



OF the people who survived females are greater in number compared to males

In [9]:
```
survivedData.describe()
```

Out[9]:

	survived	pclass	age	sibsp	parch	fare
count	342.0	342.000000	290.000000	342.000000	342.000000	342.000000
mean	1.0	1.950292	28.343690	0.473684	0.464912	48.395408
std	0.0	0.863321	14.950952	0.708688	0.771712	66.596998
min	1.0	1.000000	0.420000	0.000000	0.000000	0.000000
25%	1.0	1.000000	19.000000	0.000000	0.000000	12.475000
50%	1.0	2.000000	28.000000	0.000000	0.000000	26.000000
75%	1.0	3.000000	36.000000	1.000000	1.000000	57.000000
max	1.0	3.000000	80.000000	4.000000	5.000000	512.329200

Of the people who survived the middle 50 % of the survivors are of the age group 19 to 36 which confirms our earlier analysis through histogram

Problem2

In [10]:
```
autoData=pd.read_csv("https://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg/auto-mpg.data",delim_whites
pace=True,names=['mpg','cylinders','displacement','horsepower','weight','acceleration','model_year','origin','car_na
me'],na_values=['?'])
```

In [11]:
```
autoData.head()
```

Out[11]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	car_name
0	18.0	8	307.0	130.0	3504.0	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165.0	3693.0	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150.0	3436.0	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150.0	3433.0	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140.0	3449.0	10.5	70	1	ford torino

In [12]:
```
autoData['horsepower'].isna().value_counts()
```

Out[12]:
```
False    392
True       6
Name: horsepower, dtype: int64
```

In [13]:
```
autoData.describe()
```

Out[13]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin
count	398.000000	398.000000	398.000000	392.000000	398.000000	398.000000	398.000000	398.000000
mean	23.514573	5.454774	193.425879	104.469388	2970.424623	15.568090	76.010050	1.572864
std	7.815984	1.701004	104.269838	38.491160	846.841774	2.757689	3.697627	0.802055
min	9.000000	3.000000	68.000000	46.000000	1613.000000	8.000000	70.000000	1.000000
25%	17.500000	4.000000	104.250000	75.000000	2223.750000	13.825000	73.000000	1.000000
50%	23.000000	4.000000	148.500000	93.500000	2803.500000	15.500000	76.000000	1.000000
75%	29.000000	8.000000	262.000000	126.000000	3608.000000	17.175000	79.000000	2.000000
max	46.600000	8.000000	455.000000	230.000000	5140.000000	24.800000	82.000000	3.000000

In [14]:
```
print("The VARIANCE  before applying imputing is {0}".format(autoData['horsepower'].var()))
```

```
The VARIANCE  before applying imputing is 1481.5693929745862
```

In [15]:
```
df1=autoData.copy()
simpleImputer=SimpleImputer(missing_values=np.nan,strategy='mean')
df1['horsepower']=simpleImputer.fit_transform(df1[['horsepower']])
print("The VARIANCE after MEAN imputing is {0}".format(df1['horsepower'].var()))
```

```
The VARIANCE after MEAN imputing is 1459.1779160026776
```

In [16]:
```
df2=autoData.copy()
simpleImputer=SimpleImputer(missing_values=np.nan,strategy='median')
df2['horsepower']=simpleImputer.fit_transform(df2[['horsepower']])
print("The VARIANCE after MEDIAN imputing is {0}".format(df2['horsepower'].var()))
```

```
The VARIANCE after MEDIAN imputing is 1460.96905180816
```

In [17]:
```
df3=autoData.copy()
simpleImputer=SimpleImputer(missing_values=np.nan,strategy='most_frequent')
df3['horsepower']=simpleImputer.fit_transform(df3[['horsepower']])
print("The VARIANCE after MODE imputing is {0}".format(df3['horsepower'].var()))
```

```
The VARIANCE after MODE imputing is 1490.0361252104324
```

Mean imputing results in lowest variance as it repalces the value with the mena so the distribution of data doesnot change significantly

The other strategy is to drop the missing values and building a model to predict the missing values by giving them as query points

Problem 3

In [18]:
```
irisData=sns.load_dataset('iris')
```

In [19]:
```
type(irisData)
```

Out[19]:
```
pandas.core.frame.DataFrame
```

In [20]:
```
irisData.shape
```

Out[20]:
```
(150, 5)
```

In [21]:
```
irisData.head()
```

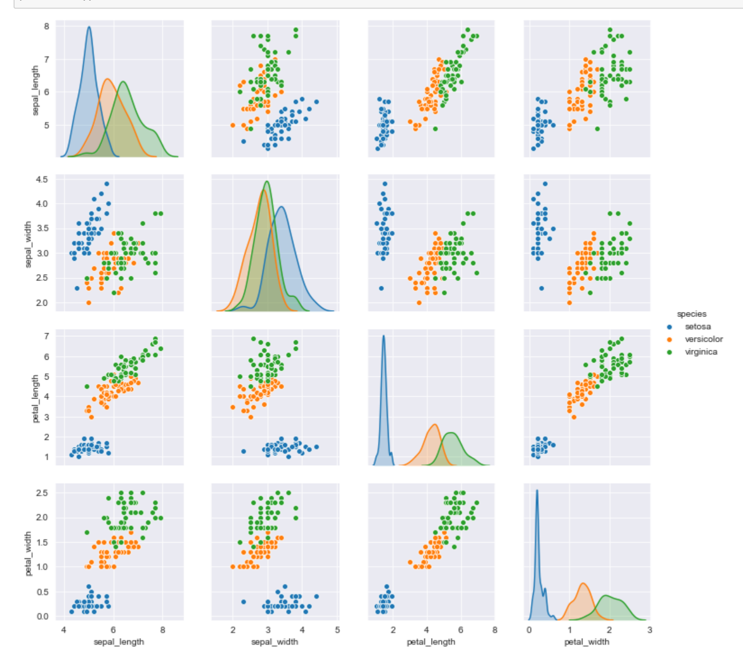
Out[21]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

In [22]:
```
irisData.dtypes
```

Out[22]:
```
sepal_length    float64
sepal_width     float64
petal_length    float64
petal_width     float64
species          object
dtype: object
```

In [23]:
```
sns.pairplot(data=irisData,hue='species')
plt.show()
```



In [24]:
```
pca=PCA(n_components=3)
pca.fit_transform(irisData.iloc[:,:-1])
print(pca.explained_variance_ratio_)
```

```
[0.92461872 0.05306648 0.01710261]
```

In [25]:
```
allVariance=list()
for column in range(len(irisData.columns)-1):
    allVariance.append(irisData.iloc[:,column].var())
    print("Variance of the {0} is {1}".format(irisData.columns[column],irisData.iloc[:,column].var()))
print("Total Variance is {0}".format(sum(allVariance)))
```

```
Variance of the sepal_length is 0.6856935123042505
Variance of the sepal_width is 0.1899794183445188
Variance of the petal_length is 3.1162778523489942
Variance of the petal_width is 0.5810062639821029
Total Variance is 4.572957046979867
```

In [26]:
```
for column in range(len(allVariance)):
    print("percentage of variance expalined by FEATURE {0} is {1}".format(irisData.columns[column],allVariance[colum
n]/sum(allVariance)))
```

```
percentage of variance expalined by FEATURE sepal_length is 0.14994532099467353
percentage of variance expalined by FEATURE sepal_width is 0.04154410732328823
percentage of variance expalined by FEATURE petal_length is 0.681457931997653
percentage of variance expalined by FEATURE petal_width is 0.12705263968438513
```

From the features the petal_length has maximum variance followed by petal_width.so these two are the important features to classify the Irsis flowers this can be seen through the pair plot

In [27]:
```
for i in range(len(pca.explained_variance_ratio_)):
    print("percentage variance expalined by principal component {0} is {1}".format(i+1,pca.explained_variance_ratio_
[i]) )
```

```
percentage variance expalined by principal component 1 is 0.9246187232017271
percentage variance expalined by principal component 2 is 0.053066483117067825
percentage variance expalined by principal component 3 is 0.017102609807929766
```

The Principal component 1 has maximum variance i.e. 92.4% followed by principal component 2 i.e. 5.3% and the principlal component 3 has least variance and explains 1% of the total data.

The max vauriance expalined by the features is 68.14 % by the Petal_length where as the Principal component 1 expalins 92.4%.So from this we can say that the Principal component 1 explains more about the data than the petal_length.

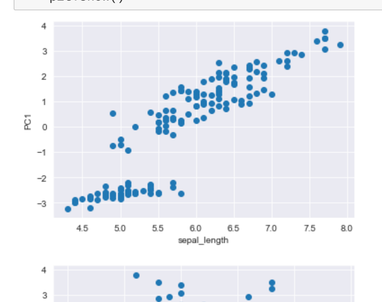
Problem 4

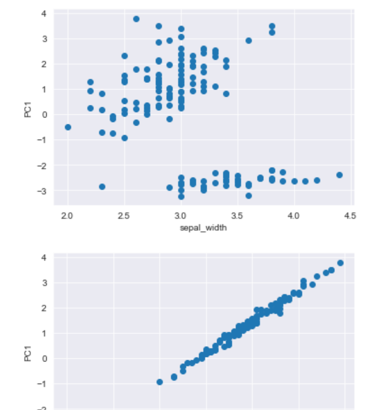
In [28]:
```
pcaDataPoints=pd.DataFrame(pca.fit_transform(irisData.iloc[:,:-1]))
```

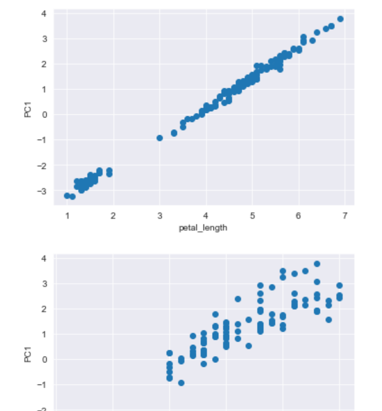
In [29]:
```
pcaDataPoints[0]
```

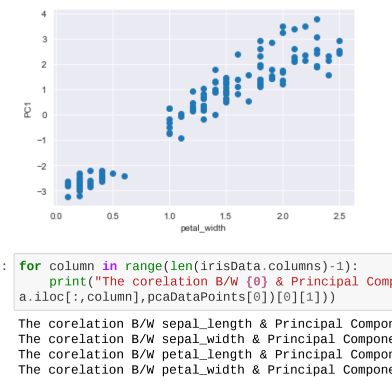
Out[29]:
```
0     -2.684126
1     -2.714142
2     -2.888991
3     -2.745343
4     -2.728717
         ...
145    1.944110
146    1.527167
147    1.764346
148    1.900942
149    1.390189
Name: 0, Length: 150, dtype: float64
```

In [30]:
```
for column in range(len(irisData.columns)-1):
    plt.scatter(x=irisData.iloc[:,column],y=pcaDataPoints[0])
    plt.xlabel(irisData.columns[column])
    plt.ylabel("PC1")
    plt.show()
```









In [31]:
```
for column in range(len(irisData.columns)-1):
    print("The corelation B/W {0} & Principal Component1 is {1}".format(irisData.columns[column],np.corrcoef(irisDat
a.iloc[:,column],pcaDataPoints[0])[0][1]))
```

```
The corelation B/W sepal_length & Principal Component1 is 0.8974017619582983
The corelation B/W sepal_width & Principal Component1 is -0.3987484724557002
The corelation B/W petal_length & Principal Component1 is 0.9978739422413107
The corelation B/W petal_width & Principal Component1 is 0.9665475167033069
```

We can see the petal_length has highest corelation coefficient followed by petal_width,sepal_length,sepal_width.
We can observe that the sepal width has inverse relation.
The above scattter plot show the same i.e. higher the corelation closer the points

Problem 5

In [32]:
```
pca1=PCA(n_components=4)
pca1.fit_transform(irisData.iloc[:,:-1])
pcaDataPoints1=pd.DataFrame(pca1.fit_transform(irisData.iloc[:,:-1]))
pca1.components_
```

Out[32]:
```
array([[ 0.36138659, -0.08452251,  0.85667061,  0.3582892 ],
       [ 0.65658877,  0.73016143, -0.17337266, -0.07548102],
       [-0.58202985,  0.59791083,  0.07623608,  0.54583143],
       [-0.31548719,  0.3197231 ,  0.47983899, -0.75365743]])
```

In [33]:
```
for column in range(len(allVariance)):
    print("percentage of variance expalined by FEATURE {0} is {1}".format(irisData.columns[column],allVariance[colum
n]/sum(allVariance)))
```

```
percentage of variance expalined by FEATURE sepal_length is 0.14994532099467353
percentage of variance expalined by FEATURE sepal_width is 0.04154410732328823
percentage of variance expalined by FEATURE petal_length is 0.681457931997653
percentage of variance expalined by FEATURE petal_width is 0.12705263968438513
```

In [34]:
```
allVariancePCA=list()
for column in range(len(pcaDataPoints1.columns)-1):
    allVariancePCA.append(pcaDataPoints1.iloc[:,column].var())
    print("Variance of the {0} is {1}".format(pcaDataPoints1.columns[column],pcaDataPoints1.iloc[:,column].var()))
print("Total Variance is {0}".format(sum(allVariancePCA)))
```

```
Variance of the 0 is 4.228241706034864
Variance of the 1 is 0.24267074792863336
Variance of the 2 is 0.07820950004291939
Total Variance is 4.5491219540064165
```

In [35]:
```
allVariance=list()
for column in range(len(irisData.columns)-1):
    allVariance.append(irisData.iloc[:,column].var())
    print("Variance of the {0} is {1}".format(irisData.columns[column],irisData.iloc[:,column].var()))
print("Total Variance is {0}".format(sum(allVariance)))
```

```
Variance of the sepal_length is 0.6856935123042505
Variance of the sepal_width is 0.1899794183445188
Variance of the petal_length is 3.1162778523489942
Variance of the petal_width is 0.5810062639821029
Total Variance is 4.572957046979867
```

In [36]:
```
for i in range(len(pca1.explained_variance_ratio_)):
    print("percentage variance expalined by principal component {0} is {1}".format(i+1,pca1.explained_variance_ratio
_[i]) )
```

```
percentage variance expalined by principal component 1 is 0.9246187232017271
percentage variance expalined by principal component 2 is 0.053066483117067825
percentage variance expalined by principal component 3 is 0.017102609807929766
percentage variance expalined by principal component 4 is 0.005212183873275377
```

If we use 4 principal components it explains almost same amount of variance as that of the original features.
To explain 95 % of the variance of data we need only Principal component 1 & Principal component 2.
As Principal component 1 & Principal component 2 expalins 0.9776 i.e. 97.76% of the data.
This implies we only need two principal components.

Recitation Exercises

Chapter 1

Question 1

Data mining is the process of automatically discovering useful information in large data.Data mining techniques allows us to find useful patterns that might otherwise remain unknown if tried manually.

1) No.It doesnt need Data Mining techniques as it can be found querying database.
2) No.It doesnt need Data Mining techniques as it can be found querying database.
3) No.It doesnt need Data Mining techniques as it can be found querying database.
4) No.It doesnt need Data Mining techniques as it can be found querying database.
5) No.It doesnt need Data Mining techniques as it can be found using mathenatical calulations.
6) yes.the outcomes of future stock price needs substanstial amount of application of Data Mining Techniques
7) yes.The predictions of abnormalities needs substanstial amount of application of Data Mining Techniques
8) yes.The predictions of earthquakes needs substanstial amount of application of Data Mining Techniques on seismic waves data.
9) No.It does not need data mining techniques as it can be done using basic signal processing

Chapter 2

Question 2

1) **Time AM PM**:: Binary, qualitative, ordinal.
2) **Brightness light meter**:: Continuous quantitative ratio.
3) **Brightness people**:: Discrete qualitative ordinal.
4) **Angles 0 to 360**::Continuous quantitative ratio.
5) **Medals Olympics**::Discreet qualitative ordinal.
6) **Height Sea level** :: Continuous quantitative ratio.
7) **patients** :: Discreet quantitative ratio.
8) **ISBN** :Discreet qualitative ordinal.
9) **pass light** :: Discreet qualitative ordinal.
10) **Military rank** :: Discreet qualitative ordinal.
11) **Distance from campus** :: Continuous quantitative ratio.
12) **Density** :: Continuous quantitative ratio.
13) **Coat check number**:: Discreet qualitative ordinal.

Question 7

Daily temperature has more autocorrealtion.It deals uses the concept of temporal autocorrelation whic says, if two measurmenrts are close in time then the values of those measurments are often very similar.In realistic scenario temperature across closer locations are very similar where as rainfall is not.

Question 15

1) We get equal number of elements from each group.
2) We dont gaurantee any specfic number from each group the elemnets are picked randomly.

Question 16

1) Term that occurs in every document has very less weight or '0' weight
2) Term that occurs only once has very high weight or maximum weight

Question 17

1) (A, B) = (a^2,b^2)
2) y=x^2

Question 18

1)
x=0101010001
y=0100011000
Jaccard Distance:2/(10-5)=2/5
Hamming Distance:3
2)
Simple Matching Coefficient and Hamming distance are equivalent.As the hamming distance gives how many diffrent points are present.Where as Simple Matching Coefficient gives the number of same attributes divided by total number of data points.
Cosine is similar to Jaccard.As both look one type of attributes in the entire data
3)
jacard distance seems more appropriate as it can show the similarity or common genes between the points
4)
Hamming distance as we want to know the diffrence betwenen two genes of humans to find out.

In [86]:
```
points1=[[1,1,1,1],[0,1,0,1],[0,-1,0,1],[1,1,0,1,0,1],[2,-1,0,2,0,-3]]
points2=[[2,2,2,2],[1,0,1,0],[1,0,-1,0],[1,1,1,0,0,1],[-1,1,-1,0,0,-1]]
ditancetype=['E','EJ','E','J','']
for i in range(len(points1)):
    array1=np.array(points1[i])
    array2=np.array(points2[i])
    print("calculating the Below metrics for points {0} and {1}".format(array1,array2))
    print("\n")
    if ditancetype[i]=='E':
        print("The euclidian Distance is {0}".format(np.linalg.norm(array1-array2)))
    elif ditancetype[i]=='J':
        print("The jaccard value is {0}".format(jaccard_score(array1,array2)))
    elif ditancetype[i]=='EJ':
        print("The euclidian Distance is {0}".format(np.linalg.norm(array1-array2)))
        print("The jaccard value is {0}".format(jaccard_score(array1,array2)))
    print("********************************")
    print("The Corelation Coefficient is  {0}".format(np.corrcoef(array1,array2)[0][1]))
    #print(np.corrcoef(array1,array2)[0][1])
    print("********************************")
    unitVector1= points1[i] / np.linalg.norm(points1[i])
    unitVector2 = points2[i] / np.linalg.norm(points2[i])
    dotPrduct=np.dot(unitVector1, unitVector2)
    x=np.arccos(dotPrduct)
    print("The Cosine Similarity  is {0} and angle between the points is {1}".format(scipy.spatial.distance.cosine(a
rray1,array2),
                                                                                                     math.cos(x)))
    print("********************************")
    print("\n")
```

```
calculating the Below metrics for points [1 1 1 1] and [2 2 2 2]


The euclidian Distance is 2.0
********************************
The Corelation Coefficient is  nan
********************************
The Cosine Similarity  is 0.0 and angle between the points is 1.0
********************************


calculating the Below metrics for points [0 1 0 1] and [1 0 1 0]


The euclidian Distance is 2.0
The jaccard value is 0.0
********************************
The Corelation Coefficient is  -1.0
********************************
The Cosine Similarity  is 1.0 and angle between the points is 6.123233995736766e-17
********************************


calculating the Below metrics for points [ 0 -1  0  1] and [ 1  0 -1  0]


The euclidian Distance is 2.0
********************************
The Corelation Coefficient is  0.0
********************************
The Cosine Similarity  is 1.0 and angle between the points is 6.123233995736766e-17
********************************


calculating the Below metrics for points [1 1 0 1 0 1] and [1 1 1 0 0 1]


The jaccard value is 0.6
********************************
The Corelation Coefficient is  0.25000000000000006
********************************
The Cosine Similarity  is 0.25 and angle between the points is 0.75
********************************


calculating the Below metrics for points [ 2 -1  0  2  0 -3] and [-1  1 -1  0  0 -1]


********************************
The Corelation Coefficient is  -2.866583523299505e-17
********************************
The Cosine Similarity  is 1.0 and angle between the points is 6.123233995736766e-17
********************************
```

Question 19

From the above we can see for points

1) x = (1, 1, 1, 1), y = (2, 2, 2, 2)
cosine = 1.0
correlation =nan
Euclidean =2.0

2) x = (0, 1, 0, 1), y = (1, 0, 1, 0)
cosine = 0 as(6.123233995736766e-17 is very negligible value)
correlation = -1.0
Euclidean = 2.0
Jaccard = 0.0

3) x=(0, −1, 0, 1), y=(1, 0, −1, 0)
cosine = 0 as(6.123233995736766e-17 is very negligible value)
correlation = 0.0
Euclidean = 2.0

4) x=(1, 1, 0, 1, 0, 1), y=(1, 1, 1, 0, 0, 1)
cosine = 0.75
correlation = 0.25
Jaccard = 0.6

5) x=(2, −1, 0, 2, 0, −3), y=(−1, 1, −1, 0, 0, −1)
cosine = 0 as(6.123233995736766e-17 is very negligible value)
correlation = 0 as(-2.866583523299505e-17 is very negligible value)