Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Кафедра системотехніки

Дисципліна: «Методи та системи штучного інтелекту»

ПРАКТИЧНА РОБОТА № 6

«Зменшення розмірності даних»

Виконала:
ст. гр. ІТКН-18-4
Левченко А.С.

Прийняв:
к.т.н., ст. викл. каф.СТ
Жернова П.Є.

Харків 2020

Мета роботи:

Отримання практичних навичок при обробці даних для подальшої їх візуалізації та моделювання за допомогою методу PCA та t-SNE.

Хід роботи:

1. Необхідно імпортувати необхідні пакети і класи: Pandas, Numpy, Sklearn, matplotlib.pyplot.

1. Імпорт необхідних пакетів

In [1]:
```
import sklearn as sk
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Рисунок 1 – Імпорт необхідних пакетів та класів

2. Імпортувати набір даних для подальшої роботи.

Для завантаження .csv файлу з даними в pandas використовується функція read_csv ().

2. Імпорт набору даних для подальшої роботи

In [2]:
```
dataset = pd.read_csv('Wine.csv')
dataset
```

Out[2]:

	Alcohol	Malic_Acid	Ash	Ash_Alcanity	Magnesium	Total_Phenols	Flavanoids	Nonflavanoid_Phenols	Proa
0	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	
1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	
2	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	
3	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	
4	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	
...	...	...	...	...	...	...	...	...	
173	13.71	5.65	2.45	20.5	95	1.68	0.61	0.52	

Рисунок 2 – Імпорт набору даних для подальшої роботи

3. Перевірити на наявність пропущених даних та заповнити їх. Необхідно скористатись декількома методами для заповнення пропусків.

3. Перевірка на наявність пропущених даних

```
In [6]: 1 dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 178 entries, 0 to 177
Data columns (total 14 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Alcohol               178 non-null    float64
 1   Malic_Acid            178 non-null    float64
 2   Ash                   178 non-null    float64
 3   Ash_Alcanity          178 non-null    float64
 4   Magnesium             178 non-null    int64
 5   Total_Phenols         178 non-null    float64
 6   Flavanoids            178 non-null    float64
 7   Nonflavanoid_Phenols  178 non-null    float64
 8   Proanthocyanins       178 non-null    float64
 9   Color_Intensity       178 non-null    float64
 10  Hue                   178 non-null    float64
 11  OD280                 178 non-null    float64
 12  Proline               178 non-null    int64
 13  Customer_Segment      178 non-null    int64
dtypes: float64(11), int64(3)
memory usage: 19.6 KB
```

Рисунок 3 – Перевірка на наявність пропущених даних за допомогою функції *.info()*

```
In [9]: 1 pd.isnull(dataset).any()
```

```
Out[9]: Alcohol               False
        Malic_Acid            False
        Ash                   False
        Ash_Alcanity          False
        Magnesium             False
        Total_Phenols         False
        Flavanoids            False
        Nonflavanoid_Phenols  False
        Proanthocyanins       False
        Color_Intensity       False
        Hue                   False
        OD280                 False
        Proline               False
        Customer_Segment      False
        dtype: bool
```

Рисунок 4 – Перевірка на наявність пропущених даних за допомогою функції *.isnull()* та. *any()*

Див. рис. 4, можна сказати, що у будь-якому стовпці, немає наявних пропущених даних.

Відсутні дані об'єктів можна замінити на конкретні числові значення, для цього можна використовувати метод fillna ().

In [7]:
```
coppy=dataset
coppy.fillna(0)
```

Рисунок 5 – Заповнення пропусків за допомогою ф-ції .fillna()

Дані можна заповнити середнім значенням по стовпцю.

In [8]:
```
coppy.fillna(coppy.mean())
```

Out[8]:

	Alcohol	Malic_Acid	Ash	Ash_Alcanity	Magnesium	Total_Phenols	Flavanoids	Nonflavanoid
0	14.23	1.71	2.43	15.6	127	2.80	3.06	
1	13.20	1.78	2.14	11.2	100	2.65	2.76	
2	13.16	2.36	2.67	18.6	101	2.80	3.24	
3	14.37	1.95	2.50	16.8	113	3.85	3.49	
4	13.24	2.59	2.87	21.0	118	2.80	2.69	
...	...	...	...	...	...	...	...	...
173	13.71	5.65	2.45	20.5	95	1.68	0.61	
174	13.40	3.91	2.48	23.0	102	1.80	0.75	
175	13.27	4.28	2.26	20.0	120	1.59	0.69	
176	13.17	2.59	2.37	20.0	120	1.65	0.68	
177	14.13	4.10	2.74	24.5	96	2.05	0.76	

178 rows × 14 columns

Рисунок 6 – Заповнення пропусків середніми значеннями

Досить часто використовуваний підхід при роботі з відсутніми даними - це видалення записів (рядків) або полів (стовпців), в яких зустрічаються пропуски. Для того, щоб видалити всі об'єкти, які містять значення NaN скористайтеся методом dropna () без аргументів.

```
In [9]:  1 coppy.dropna()
```

Out[9]:

	Alcohol	Malic_Acid	Ash	Ash_Alcanity	Magnesium	Total_Phenols	Flavanoids	Nonflavanoid_Phenols	Proanthocyanins
0	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	2.29
1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28
2	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	2.81
3	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	2.18
4	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	1.82
...	...	...	...	...	...	...	...	...	...
173	13.71	5.65	2.45	20.5	95	1.68	0.61	0.52	1.06
174	13.40	3.91	2.48	23.0	102	1.80	0.75	0.43	1.41
175	13.27	4.28	2.26	20.0	120	1.59	0.69	0.43	1.35
176	13.17	2.59	2.37	20.0	120	1.65	0.68	0.53	1.46
177	14.13	4.10	2.74	24.5	96	2.05	0.76	0.56	1.35

178 rows × 14 columns

Рисунок 7 – Заповнення пропусків середніми значеннями

```
In [12]:  1 X = dataset.iloc[:, :-1].values
          2 Y = dataset.iloc[:, -1].values
```

Рисунок 8 – Розподіл набору даних на X та У

```
In [10]:  1 from sklearn.impute import SimpleImputer
          2 X = dataset.iloc[:, :-1].values
          3 imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
          4 imputer.fit(X[:, :-1])
          5 X[:, :-1] = imputer.transform(X[:, :-1])
          6 print(X)
```

```
[[1.423e+01 1.710e+00 2.430e+00 ... 1.040e+00 3.920e+00 1.065e+03]
 [1.320e+01 1.780e+00 2.140e+00 ... 1.050e+00 3.400e+00 1.050e+03]
 [1.316e+01 2.360e+00 2.670e+00 ... 1.030e+00 3.170e+00 1.185e+03]
 ...
 [1.327e+01 4.280e+00 2.260e+00 ... 5.900e-01 1.560e+00 8.350e+02]
 [1.317e+01 2.590e+00 2.370e+00 ... 6.000e-01 1.620e+00 8.400e+02]
 [1.413e+01 4.100e+00 2.740e+00 ... 6.100e-01 1.600e+00 5.600e+02]]
```

Рисунок 8 – Заповнення даних за допомогою SimpleImputer, strategy="mean"

```
In [11]:  1 from sklearn.impute import SimpleImputer
          2 X = dataset.iloc[:, :-1].values
          3 imputer = SimpleImputer(missing_values=np.nan, strategy='median')
          4 imputer.fit(X[:, :-1])
          5 X[:, :-1] = imputer.transform(X[:, :-1])
          6 print(X)
```

```
[[1.423e+01 1.710e+00 2.430e+00 ... 1.040e+00 3.920e+00 1.065e+03]
 [1.320e+01 1.780e+00 2.140e+00 ... 1.050e+00 3.400e+00 1.050e+03]
 [1.316e+01 2.360e+00 2.670e+00 ... 1.030e+00 3.170e+00 1.185e+03]
 ...
 [1.327e+01 4.280e+00 2.260e+00 ... 5.900e-01 1.560e+00 8.350e+02]
 [1.317e+01 2.590e+00 2.370e+00 ... 6.000e-01 1.620e+00 8.400e+02]
 [1.413e+01 4.100e+00 2.740e+00 ... 6.100e-01 1.600e+00 5.600e+02]]
```

Рисунок 9 – Заповнення даних за допомогою SimpleImputer, strategy="median"

In [12]:

```
from sklearn.impute import SimpleImputer
X = dataset.iloc[:, :-1].values
imputer = SimpleImputer(missing_values=np.nan, strategy='most_frequent')
imputer.fit(X[:, :-1])
X[:, :-1] = imputer.transform(X[:, :-1])
print(X)
```

```
[[1.423e+01 1.710e+00 2.430e+00 ... 1.040e+00 3.920e+00 1.065e+03]
 [1.320e+01 1.780e+00 2.140e+00 ... 1.050e+00 3.400e+00 1.050e+03]
 [1.316e+01 2.360e+00 2.670e+00 ... 1.030e+00 3.170e+00 1.185e+03]
 ...
 [1.327e+01 4.280e+00 2.260e+00 ... 5.900e-01 1.560e+00 8.350e+02]
 [1.317e+01 2.590e+00 2.370e+00 ... 6.000e-01 1.620e+00 8.400e+02]
 [1.413e+01 4.100e+00 2.740e+00 ... 6.100e-01 1.600e+00 5.600e+02]]
```

Рисунок 10 – Заповнення даних за допомогою SimpleImputer, strategy="most_frequent"

За першими двома ознаками

In [18]:

```
fig = plt.figure(figsize = (8,8))
ax = fig.add_subplot(1,1,1)
ax.set_xlabel('Alcohol', fontsize = 15)
ax.set_ylabel('Malic_Acid', fontsize = 15)
ax.set_title('Data visualization', fontsize = 20)
targets = [1, 2, 3]
colors = ['r', 'g', 'b']
for target, color in zip(targets,colors):
    indicesToKeep = dataset['Customer_Segment'] == target
    ax.scatter(dataset.loc[indicesToKeep, 'Alcohol'],
               dataset.loc[indicesToKeep, 'Malic_Acid'],
               c = color,
               s = 50)
ax.legend(targets)
ax.grid()
```

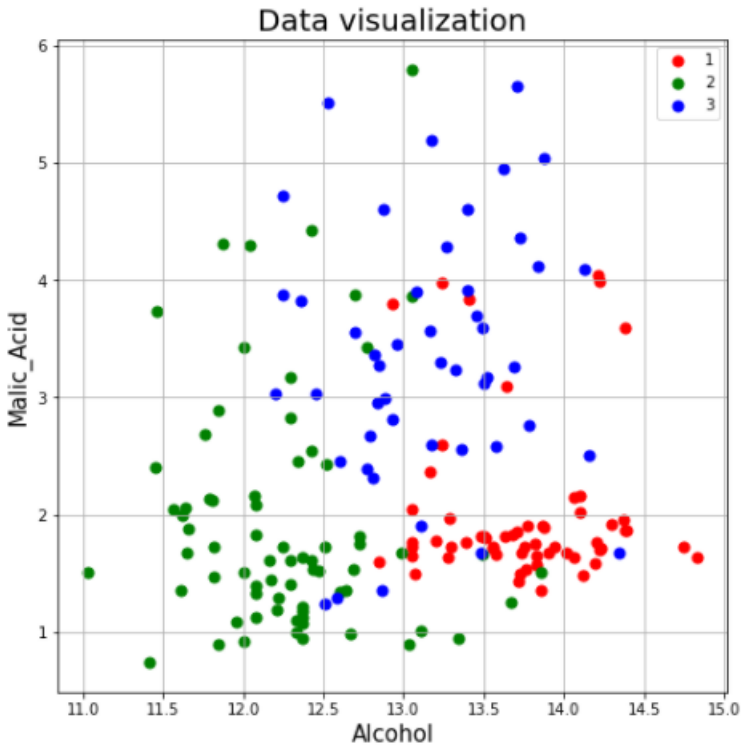


Рисунок 11 – Візуалізація початкових даних за першими двома ознаками

За першими трьома ознаками

In [14]:
```
fig = plt.figure(figsize = (8,8))
ax = Axes3D(fig, elev=-130, azim=70)
ax.set_xlabel('Alcohol', fontsize = 15)
ax.set_ylabel('Malic_Acid', fontsize = 15)
ax.set_zlabel('Ash', fontsize = 15)
ax.set_title('Data visualization 3D', fontsize = 20)
targets = [1, 2, 3]
colors = ['r', 'g', 'b']
for target, color in zip(targets,colors):
    indicesToKeep = dataset['Customer_Segment'] == target
    ax.scatter(dataset.loc[indicesToKeep, 'Alcohol'],
               dataset.loc[indicesToKeep, 'Malic_Acid'],
               dataset.loc[indicesToKeep, 'Ash'],
               edgecolor='k',
               alpha = 1,
               c = color,
               s = 50)
ax.legend(targets)
ax.grid()
```

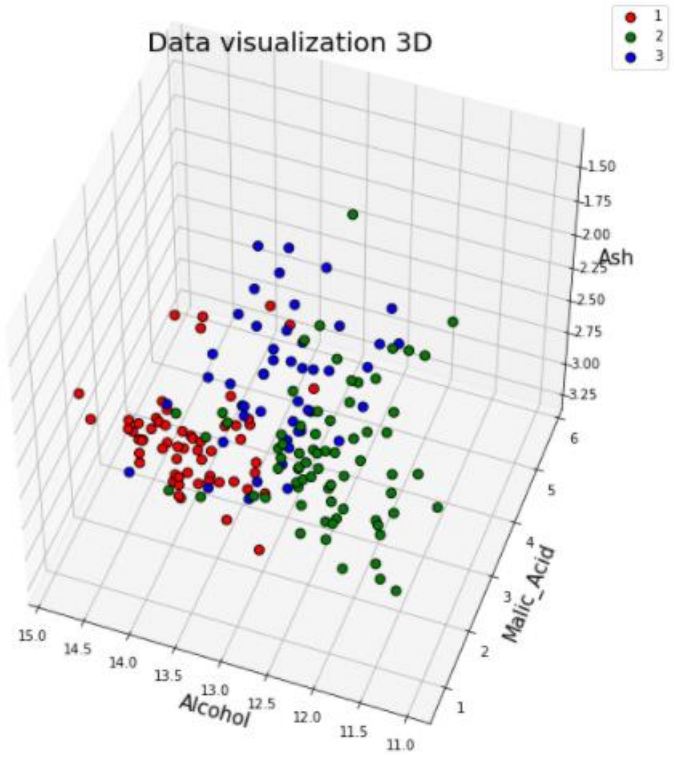


Рисунок 12 – Візуалізація початкових даних за першими трьома ознаками

4. Розділити набір даних на тестову та тренувальну вибірку даних.

In [20]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=1)
print("X_train", X_train)
print("X_test", X_test)
print("y_train", y_train)
print("y_test", y_test)
```

```
X_train [[1.200e+01 1.510e+00 2.420e+00 ... 1.050e+00 2.650e+00 4.500e+02]
 [1.272e+01 1.810e+00 2.200e+00 ... 1.160e+00 3.140e+00 7.140e+02]
 [1.208e+01 1.390e+00 2.500e+00 ... 9.300e-01 3.190e+00 3.850e+02]
 ...
 [1.349e+01 1.660e+00 2.240e+00 ... 9.800e-01 2.780e+00 4.720e+02]
 [1.293e+01 2.810e+00 2.700e+00 ... 7.700e-01 2.310e+00 6.000e+02]
 [1.305e+01 1.650e+00 2.550e+00 ... 1.120e+00 2.510e+00 1.105e+03]]
X_test [[1.369e+01 3.260e+00 2.540e+00 2.000e+01 1.070e+02 1.830e+00 5.600e-01
  5.000e-01 8.000e-01 5.880e+00 9.600e-01 1.820e+00 6.800e+02]
 [1.242e+01 1.610e+00 2.190e+00 2.250e+01 1.080e+02 2.000e+00 2.090e+00
  3.400e-01 1.610e+00 2.060e+00 1.060e+00 2.960e+00 3.450e+021
```

Рисунок 13 – Розподіл даних на тестову та тренувальну вибірки

5. Виконати масштабування даних.

Маштабування даних

```
In [23]:
from sklearn.preprocessing import MinMaxScaler
sc = MinMaxScaler(feature_range=(0,1))
sc.fit(X)
X = sc.transform(X)
print(X)
X_train = sc.transform(X_train)
X_test = sc.transform(X_test)
print("Масштаб. тренувальна вибірка X")
print(X_train)
print("Масштаб. тестова вибірка X")
print(X_test)
```

```
[[0.84210526 0.1916996  0.57219251 ... 0.45528455 0.97069597 0.56134094]
 [0.57105263 0.2055336  0.4171123  ... 0.46341463 0.78021978 0.55064194]
 [0.56052632 0.3201581  0.70053476 ... 0.44715447 0.6959707  0.64693295]
 ...
 [0.58947368 0.69960474 0.48128342 ... 0.08943089 0.10622711 0.39728959]
 [0.56315789 0.36561265 0.54010695 ... 0.09756098 0.12820513 0.40085592]
 [0.81578947 0.66403162 0.73796791 ... 0.10569106 0.12087912 0.20114123]]
Масштаб. тренувальна вибірка X
[[1.200e+01 1.510e+00 2.420e+00 ... 1.050e+00 2.650e+00 4.500e+02]
 [1.272e+01 1.810e+00 2.200e+00 ... 1.160e+00 3.140e+00 7.140e+02]
 [1.208e+01 1.390e+00 2.500e+00 ... 9.300e-01 3.190e+00 3.850e+02]
 ...
 [1.349e+01 1.660e+00 2.240e+00 ... 9.800e-01 2.780e+00 4.720e+02]
 [1.293e+01 2.810e+00 2.700e+00 ... 7.700e-01 2.310e+00 6.000e+02]
 [1.305e+01 1.650e+00 2.550e+00 ... 1.120e+00 2.510e+00 1.105e+03]]
Масштаб. тестова вибірка X
[[1.369e+01 3.260e+00 2.540e+00 2.000e+01 1.070e+02 1.830e+00 5.600e-01
  5.000e-01 8.000e-01 5.880e+00 9.600e-01 1.820e+00 6.800e+02]
 [1.242e+01 1.610e+00 2.190e+00 2.250e+01 1.080e+02 2.000e+00 2.090e+00
  3.400e-01 1.610e+00 2.060e+00 1.060e+00 2.960e+00 3.450e+02]
```

Рисунок 14 – Масштабування даних за допомогою метода *MinMaxScler()*

6. Зменшити розмірність вхідних даних за домомогою методу PCA та t-SNE.

6.1 Зменшення розмірності даних за допомогою методу PCA

Метод PCA

```
In [25]:
from sklearn.decomposition import PCA
pca_method = PCA(n_components=3)
principalComponents = pca_method.fit_transform(X)
print("Метод PCA для X з 3 компонентами")
print(principalComponents)
```

```
Метод PCA для X з 3 компонентами
[[-7.06335756e-01 -2.53192753e-01  2.40926932e-02]
 [-4.84976802e-01 -8.82289142e-03 -2.80482048e-01]
 [-5.21172266e-01 -1.89187222e-01  1.96216736e-01]
 [-8.21643663e-01 -5.80905512e-01  8.11097172e-02]
 [-2.02546382e-01 -5.94665740e-02  3.00239941e-01]
 [-6.08190152e-01 -4.87519191e-01 -7.54332321e-02]
 [-5.44047399e-01 -3.00196497e-01 -1.05074621e-01]
 [-4.74357495e-01 -2.98197021e-01 -2.82149308e-03]
 [-5.00432012e-01 -3.07602859e-01 -2.30493613e-01]
```

Рисунок 15 – Код зменшення розмірності методом PCA

In [27]:

```
principDataset = pd.DataFrame(data = principalComponents [:, 0:2],
columns = ['principal component 1', 'principal component 2'])
principDataset
finalDataset = pd.concat([principDataset, dataset[['Customer_Segment']]], axis = 1)
finalDataset
```

Out[27]:

	principal component 1	principal component 2	Customer_Segment
0	-0.706336	-0.253193	1
1	-0.484977	-0.008823	1
2	-0.521172	-0.189187	1
3	-0.821644	-0.580906	1
4	-0.202546	-0.059467	1
...	...	...	...
173	0.739510	-0.471901	3
174	0.581781	-0.348366	3
175	0.626313	-0.546857	3
176	0.572991	-0.425516	3
177	0.701764	-0.513505	3

178 rows × 3 columns

Рисунок 16 – Створення таблиці для двовимірного відображення

Візуалізація для методу PCA з 2 компонентами

In [28]:

```
fig = plt.figure(figsize = (8,8))
ax = fig.add_subplot(1,1,1)
ax.set_xlabel('Principal Component 1', fontsize = 15)
ax.set_ylabel('Principal Component 2', fontsize = 15)
ax.set_title('2 component PCA', fontsize = 20)
targets = [1, 2, 3]
colors = ['r', 'g', 'b']
for target, color in zip(targets,colors):
    indicesToKeep = finalDataset['Customer_Segment'] == target
    ax.scatter(finalDataset.loc[indicesToKeep, 'principal component 1']
               , finalDataset.loc[indicesToKeep, 'principal component 2']
               , c = color
               , s = 50)
ax.legend(targets)
ax.grid()
```

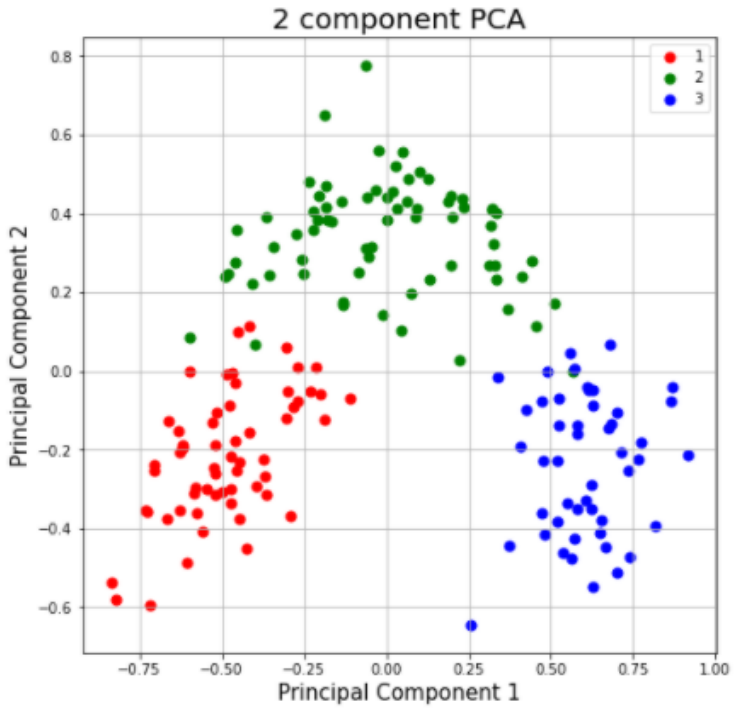


Рисунок 17 – Візуалізація для методу PCA з 2 компонентами

Формування таблиці за методом PCA для тривимірної візуалізації

In [29]:
```
principDataset3 = pd.DataFrame(data = principalComponents [:, 0:3],
columns = ['p_comp 1', 'p_comp 2','p_comp 3'])
finalDataset3 = pd.concat([principDataset3, dataset[['Customer_Segment']]], axis = 1)
finalDataset3
```

Out[29]:

	p_comp 1	p_comp 2	p_comp 3	Customer_Segment
0	-0.706336	-0.253193	0.024093	1
1	-0.484977	-0.008823	-0.280482	1
2	-0.521172	-0.189187	0.196217	1
3	-0.821644	-0.580906	0.081110	1
4	-0.202546	-0.059467	0.300240	1
...	...	...	...	...
173	0.739510	-0.471901	0.209360	3
174	0.581781	-0.348366	0.083590	3
175	0.626313	-0.546857	-0.030495	3
176	0.672991	-0.425516	-0.094537	3
177	0.701764	-0.513505	0.293910	3

178 rows × 4 columns

Рисунок 18 – Створення таблиці для трьохвимірного відображення

In [30]:
```
ax = Axes3D(fig, elev=-170, azim=70)
ax.set_xlabel('Principal Component 1', fontsize = 15)
ax.set_ylabel('Principal Component 2', fontsize = 15)
ax.set_zlabel('Principal Component 3', fontsize = 15)
ax.set_title('3 component PCA', fontsize = 20)
targets = [1, 2, 3]
colors = ['r', 'g', 'b']
for target, color in zip(targets,colors):
    indicesToKeep = finalDataset3['Customer_Segment'] == target
    ax.scatter(finalDataset3.loc[indicesToKeep, 'p_comp 1'],
               finalDataset3.loc[indicesToKeep, 'p_comp 2'],
               finalDataset3.loc[indicesToKeep, 'p_comp 3'],
               c = color, alpha = 1, edgecolor='k',
               s = 50)
ax.legend(targets)
ax.grid()
```

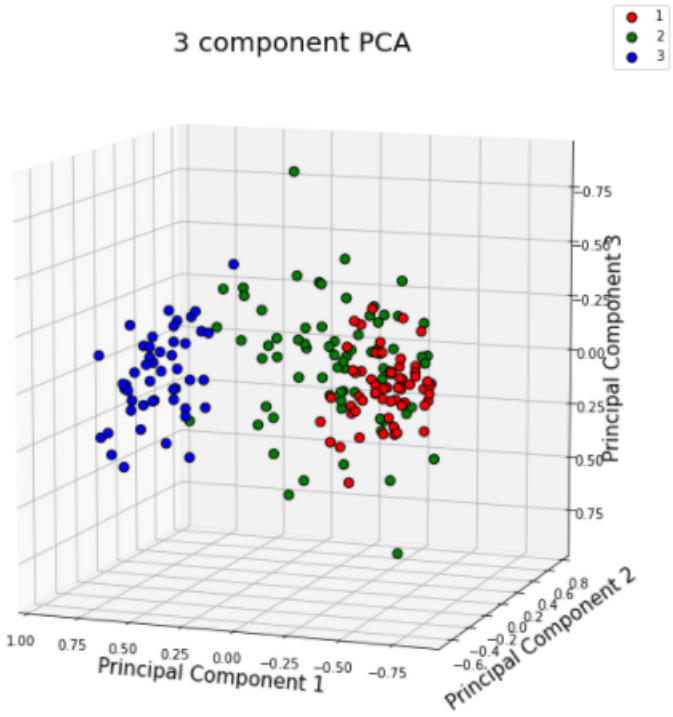


Рисунок 19 – Візуалізація для методу PCA з 3 компонентами

6.1 Зменшення розмірності даних за допомогою методу t-SNE

Метод t-SNE

In [35]:
```
from sklearn.manifold import TSNE
tsne_method = TSNE(n_components = 2, perplexity = 30, n_iter=1000, random_state = 0, learning_rate = 100)
X_2D = tsne_method.fit_transform(X)
print(X_2D)
```

```
[[  7.7138658    9.659583  ]
 [  6.4417953    6.9146867 ]
 [ 10.179637     6.9014955 ]
 [ 11.261244     9.309397  ]
 [  8.651131     2.264675  ]
 [ 11.721351     8.05081   ]
 [  8.364414     6.755891  ]
 [  9.769156     4.5484695 ]
 [  8.889901     7.4986067 ]
 [  8.705028     8.424088  ]
```

Рисунок 20 – Код зменшення розмірності методом t-SNE

Таблиця для двовимірного відображення

In [38]:
```
tsneDataset = pd.DataFrame(data = X_2D, columns = ['dim 1', 'dim 2'])
tsneDataset
final_tsneDataset = pd.concat([tsneDataset, dataset[['Customer_Segment']]], axis = 1)
final_tsneDataset
```

Out[38]:

	dim 1	dim 2	Customer_Segment
0	7.713866	9.659583	1
1	6.441795	6.914687	1
2	10.179637	6.901495	1
3	11.261244	9.309397	1
4	8.651131	2.264675	1
...	...	...	...
173	-11.864552	-10.053349	3
174	-10.085733	-10.449601	3
175	-9.606742	-12.125190	3
176	-10.115596	-12.029475	3
177	-11.450240	-11.269584	3

178 rows × 3 columns

Рисунок 21 – Створення таблиці для двовимірного відображення

Візуалізація для методу t-SNE з 2 компонентами

In [39]:
```
fig = plt.figure(figsize = (8,8))
ax = fig.add_subplot(1,1,1)
ax.set_xlabel('dim 1', fontsize = 15)
ax.set_ylabel('dim 2', fontsize = 15)
ax.set_title('2D TSNE', fontsize = 20)
targets = [1, 2, 3]
colors = ['r', 'g', 'b']
for target, color in zip(targets,colors):
    indicesToKeep = final_tsneDataset['Customer_Segment'] == target
    ax.scatter(final_tsneDataset.loc[indicesToKeep, 'dim 1']
               , final_tsneDataset.loc[indicesToKeep, 'dim 2']
               , c = color
               , s = 50)
ax.legend(targets)
ax.grid()
```

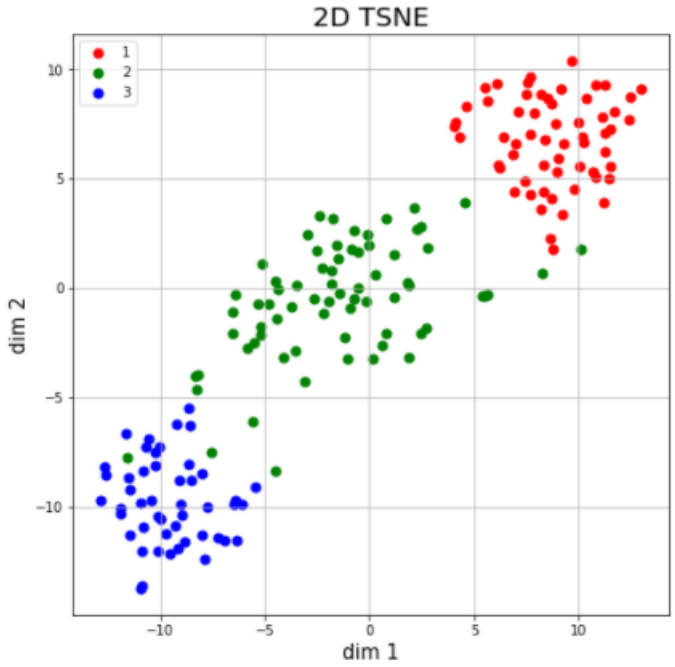


Рисунок 22 – Візуалізація для методу t-SNE з 2 компонентами

Формування таблиці за методом t-SNE для тривимірної візуалізації

In [42]:
```
tsne_method = TSNE (n_components = 3, perplexity = 5, n_iter=5000, random_state = 0, learning_rate = 100)
X_3D = tsne_method.fit_transform(X)
tsneDataset = pd.DataFrame(data = X_3D, columns = ['dim 1', 'dim 2', 'dim 3'])
tsneDataset
final_tsneDataset = pd.concat([tsneDataset, dataset[['Customer_Segment']]], axis = 1)
final_tsneDataset
```

Out[42]:

	dim 1	dim 2	dim 3	Customer_Segment
0	-102.573128	-25.631596	33.992153	1
1	-49.533241	-24.351484	14.020737	1
2	-60.553822	-14.767481	57.521206	1
3	-79.285988	-39.363842	55.620331	1
4	-52.520065	-3.574616	31.077799	1
...	...	...	...	...
173	122.854843	-25.579245	7.969127	3
174	112.313927	-27.514774	-16.957693	3
175	113.415581	-36.078808	-29.829729	3
176	118.815239	-40.796215	-24.808651	3
177	125.387894	-28.788553	-13.235101	3

178 rows × 4 columns

Рисунок 23 – Створення таблиці для трьохвимірного відображення

Візуалізація для методу t-SNE з 3 компонентами

In [41]:

```
ax = Axes3D(fig, elev=-160, azim=70)
ax.set_xlabel('dim 1', fontsize = 15)
ax.set_ylabel('dim 2', fontsize = 15)
ax.set_zlabel('dim 3', fontsize = 15)
ax.set_title('t-SNE', fontsize = 20)
targets = [1, 2, 3]
colors = ['r', 'g', 'b']
for target, color in zip(targets,colors):
    indicesToKeep = final_tsneDataset['Customer_Segment'] == target
    ax.scatter(final_tsneDataset.loc[indicesToKeep, 'dim 1'],
               final_tsneDataset.loc[indicesToKeep, 'dim 2'],
               final_tsneDataset.loc[indicesToKeep, 'dim 3'],
               c = color, alpha = 1,
               s = 50)
ax.legend(targets)
ax.grid()
```

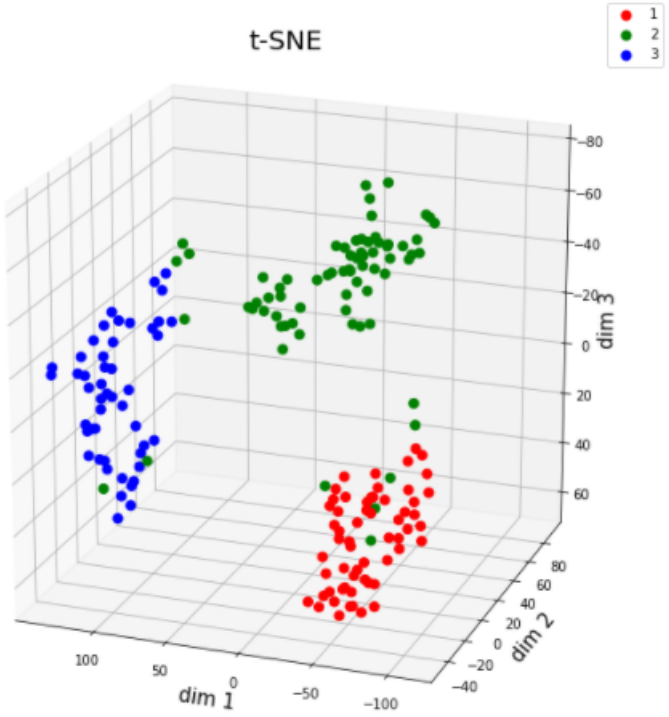
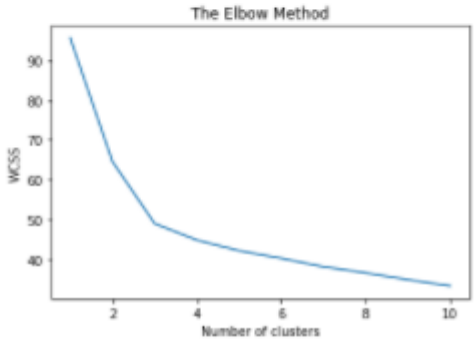


Рисунок 24 – Візуалізація для методу t-SNE з 3 компонентами

7. Провести навчання моделей на навчальному наборі даних використовуючи будь який метод.

Навчання моделі K-Means на наборі даних

In [26]:
```
from sklearn.cluster import KMeans
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```



In [27]:
```
kmeans = KMeans(n_clusters = 3, init = 'k-means++', random_state = 42)
y_kmeans = kmeans.fit_predict(X)
```

In [28]:
```
plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], s = 50, c = 'red', label = 'Cluster 1')
plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], s = 50, c = 'blue', label = 'Cluster 2')
plt.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], s = 50, c = 'green', label = 'Cluster 3')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 100, alpha = 0.9, c = 'yellow', label = 'Cent
plt.title('Clusters of customers')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.show()
```

In [29]:
```
from sklearn. metrics import confusion_matrix, accuracy_score
y_pred = kmeans.predict(X_test)
conf_matrix = confusion_matrix(y_test, y_pred)
print("Матриця відповідей:")
print(conf_matrix)
print('Оцінка точності на тестовому наборі:')
first_accur = accuracy_score(y_test,y_pred)
print("{:.0%}".format(first_accur))
```

```
Матриця відповідей:
[[ 0  0  0  0]
 [ 0 14  0  0]
 [ 0  1 12  0]
 [ 9  0  0  0]]
Оцінка точності на тестовому наборі:
72%
```

Рисунок 25 – Навчання моделей методом K-Means

Навчання моделі Random Forest на наборі даних

In [30]:
```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators=20)
rf.fit(X_train, y_train)
```

Out[30]:
```
RandomForestClassifier(n_estimators=20)
```

Прогнозування результатів тестового набору

In [31]:
```
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))
```

```
[[3 3]
 [2 2]
 [1 1]
 [2 2]
 [1 1]
 [3 3]
 [2 2]
 [1 1]
 [3 3]
 [2 2]
 [1 1]
 [1 1]
 [2 2]
 [1 1]
 [2 2]
 [2 2]
 [3 3]
 [1 1]
 [2 2]
 [1 1]
 [1 1]
 [2 2]
 [3 3]
 [1 2]
 [1 1]
 [3 3]
 [1 1]
 [1 1]
 [1 1]
 [3 3]
 [2 2]
 [3 3]
 [3 3]
 [1 1]
 [2 2]
 [2 2]]
```

In [32]:
```
conf_matrix = confusion_matrix(y_test, y_pred)
print("Матриця відповідей:")
print(conf_matrix)
print('Оцінка точності на тестовому наборі:')
print("{:.0%}".format(accuracy_score(y_test,y_pred)))
second_accur = accuracy_score(y_test,y_pred)
print(second_accur)
```

```
Матриця відповідей:
[[14  0  0]
 [ 1 12  0]
 [ 0  0  9]]
Оцінка точності на тестовому наборі:
97%
0.9722222222222222
```

Рисунок 26 – Навчання моделей методом Random Forest

Навчання моделі SVM на навчальному наборі

In [33]:
```
from sklearn.svm import SVC
classifier = SVC()
classifier.fit(X_train, y_train)
svcPred = classifier.predict(X_test)
```

In [34]:
```
from sklearn import svm
svc = svm.SVC(kernel='linear', C=1,gamma='auto').fit(X_train,y_train)
y_predicted= svc.predict(X_test)
print(y_predicted)
```

```
[3 2 1 2 1 3 2 1 3 2 1 1 2 1 2 2 3 1 2 1 1 2 3 2 1 3 1 1 1 3 2 3 3 1 2 2]
```

In [35]:
```
from matplotlib.colors import ListedColormap
result_svm = confusion_matrix(y_test, y_predicted)
print("Confusion Matrix:")
print(result_svm)
third_accur = accuracy_score(y_test,y_predicted)
print("Accuracy:",third_accur)
```

```
Confusion Matrix:
[[14  0  0]
 [ 0 13  0]
 [ 0  0  9]]
Accuracy: 1.0
```

Рисунок 27 – Навчання моделей методом SVM

8. Порівняння точності методів навчання моделей

Порівняння точності кожної моделі

In [36]:
```
Accuracy = [first_accur, second_accur, third_accur]
Methods = [' K-Means', 'Random Forest', 'SVM']
Accuracy_pos = np.arange(len(Methods))
plt.bar(Accuracy_pos, Accuracy)
plt.xticks(Accuracy_pos, Methods)
plt.title('Порівняння точності кожної моделі')
plt.show()
```

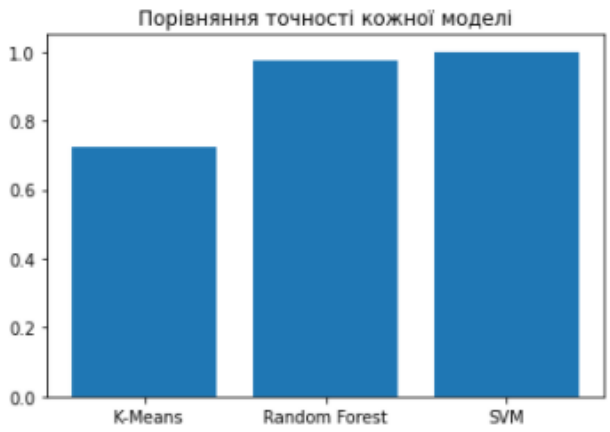


Рисунок 27 – Порівняння моделей

Висновки:

В ході виконання практичної роботи було отримано практичні навички при обробці даних для подальшої візуалізації та моделювання за допомогою методів PCA та t-SNE. Було виконано моделювання та візуалізація вхідних даних та перевірено точність цих методів методом K-Means, Random Forest, SVM. Визначили, що метод SVM в даному випадку буде мати найбільшу точність.