Name: Navneet Yadav

Roll no. 2301560044

Program: MCA-I

Subject: AI-ML

Question 2

```
GitHub: https://github.com/navYadav20/AI-ML
```

KNN classification

In [2]:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.metrics import accuracy_score
```

In [3]:

```
df = pd.read_csv('D:/ai-ml assignment/assignment 4/cancer.csv')
df.head(5)
```

Out[3]:

	id	clump_thickness	unif_cell_size	unif_cell_shape	marg_adhesion	single_epith_cel
0	1000025	5	1	1	1	
1	1002945	5	4	4	5	
2	1015425	3	1	1	1	
3	1016277	6	8	8	1	
4	1017023	4	1	1	3	

In [4]:

```
df.replace('?', -99999, inplace=True)
df.drop(columns=['id'], inplace=True)
df
```

Out[4]:

	clump_thickness	unif_cell_size	unif_cell_shape	marg_adhesion	single_epith_cell_size
0	5	1	1	1	2
1	5	4	4	5	7
2	3	1	1	1	2
3	6	8	8	1	3
4	4	1	1	3	2
...	...	...	...	...	...
694	3	1	1	1	3
695	2	1	1	1	2
696	5	10	10	3	7
697	4	8	6	4	3
698	4	8	8	5	4

699 rows × 10 columns

In [5]:

```
X = np.array(df.drop('classes', axis=1))  # Dropping 'classes' column to cr
X
```

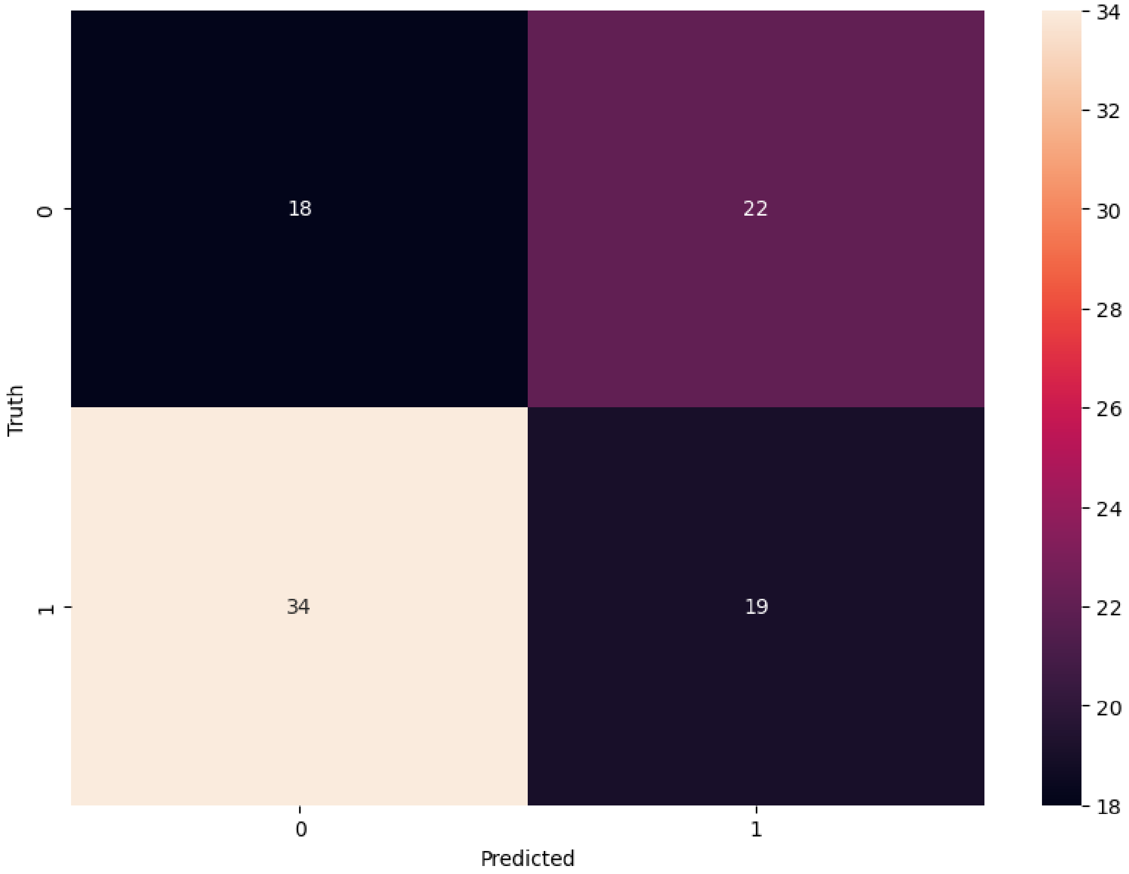
Out[5]:

```
array([[5, 1, 1, ..., 3, 1, 1],
       [5, 4, 4, ..., 3, 2, 1],
       [3, 1, 1, ..., 3, 1, 1],
       ...,
       [5, 10, 10, ..., 8, 10, 2],
       [4, 8, 6, ..., 10, 6, 1],
       [4, 8, 8, ..., 10, 4, 1]], dtype=object)
```

```
In [6]: y = np.array(df['classes'])  # Assigning the 'classes' column to the target
        y
```

```
Out[6]: array([0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1,
               0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1,
               1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1,
               0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1,
               0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1,
               0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0,
               1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0,
               0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1,
               0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0,
               0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0,
               0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0,
               0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1,
               1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1,
               1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0,
               1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1,
               1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0,
               0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0,
               0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0,
               0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0,
               0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0,
               1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0,
               0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1,
               0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
               1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0,
               0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1,
               0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1,
               0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0,
               1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0,
               0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0,
               0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
               0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
               0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1], dtype=int64)
```

In [7]:
```
import seaborn as sns

cm = [[18, 22], [ 34, 19]]
plt.figure(figsize=(10,7))
sns.heatmap(cm, annot=True, fmt='d')
plt.xlabel('Predicted')
plt.ylabel('Truth')
plt.show()
```



In [8]:
```
# Splitting the dataset into the Training set and Test set

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.35,
```

In [9]:
```
# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

In [10]:

```
#principle component analysis

from sklearn.decomposition import PCA
pca = PCA(n_components=2)
X_train = pca.fit_transform(X_train)
X_test = pca.fit_transform(X_test)
explained_variance=pca.explained_variance_ratio_
```

In [11]:

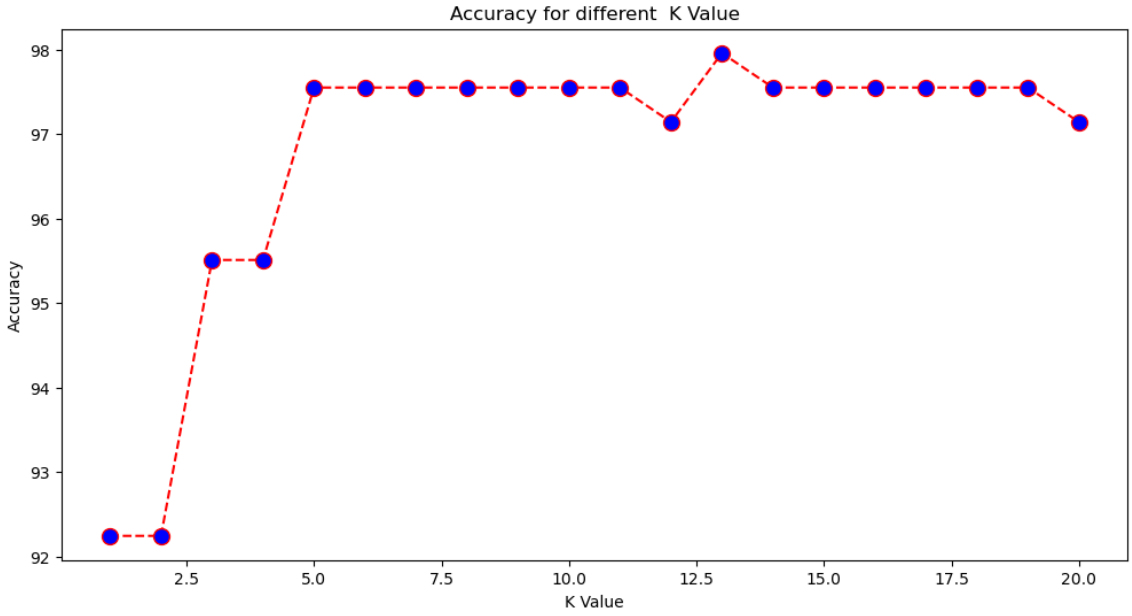
```
from sklearn.neighbors import KNeighborsClassifier
knn = []
for i in range(1,21):

    classifier = KNeighborsClassifier(n_neighbors=i)
    trained_model=classifier.fit(X_train,y_train)
    trained_model.fit(X_train,y_train )

    # Predicting the Test set results

    y_pred = classifier.predict(X_test)

    # Making the Confusion Matrix

    from sklearn.metrics import confusion_matrix

    cm_KNN = confusion_matrix(y_test, y_pred)
    print(cm_KNN)
    print("Accuracy score of train KNN")
    print(accuracy_score(y_train, trained_model.predict(X_train))*100)

    print("Accuracy score of test KNN")
    print(accuracy_score(y_test, y_pred)*100)

    knn.append(accuracy_score(y_test, y_pred)*100)
```

```
[[154  10]
 [  9  72]]
Accuracy score of train KNN
100.0
Accuracy score of test KNN
92.24489795918367
[[160   4]
 [ 15  66]]
Accuracy score of train KNN
97.79735682819384
Accuracy score of test KNN
92.24489795918367
[[160   4]
 [  7  74]]
Accuracy score of train KNN
96.91629955947137
Accuracy score of test KNN
95.51020408163265
[[160   4]
 [  7  74]]
Accuracy score of train KNN
96.47577092511013
Accuracy score of test KNN
95.51020408163265
[[160   4]
 [  2  79]]
Accuracy score of train KNN
96.69603524229075
Accuracy score of test KNN
97.55102040816327
[[160   4]
 [  2  79]]
Accuracy score of train KNN
96.69603524229075
Accuracy score of test KNN
97.55102040816327
[[160   4]
 [  2  79]]
Accuracy score of train KNN
96.69603524229075
Accuracy score of test KNN
97.55102040816327
[[160   4]
 [  2  79]]
Accuracy score of train KNN
96.91629955947137
Accuracy score of test KNN
97.55102040816327
[[160   4]
 [  2  79]]
Accuracy score of train KNN
96.69603524229075
Accuracy score of test KNN
97.55102040816327
[[160   4]
 [  2  79]]
Accuracy score of train KNN
96.69603524229075
Accuracy score of test KNN
97.55102040816327
[[160   4]
```

```
 [  2  79]]
Accuracy score of train KNN
96.69603524229075
Accuracy score of test KNN
97.55102040816327
[[160   4]
 [  3  78]]
Accuracy score of train KNN
96.69603524229075
Accuracy score of test KNN
97.14285714285714
[[160   4]
 [  1  80]]
Accuracy score of train KNN
96.25550660792952
Accuracy score of test KNN
97.95918367346938
[[160   4]
 [  2  79]]
Accuracy score of train KNN
96.25550660792952
Accuracy score of test KNN
97.55102040816327
[[160   4]
 [  2  79]]
Accuracy score of train KNN
96.25550660792952
Accuracy score of test KNN
97.55102040816327
[[160   4]
 [  2  79]]
Accuracy score of train KNN
96.25550660792952
Accuracy score of test KNN
97.55102040816327
[[160   4]
 [  2  79]]
Accuracy score of train KNN
96.25550660792952
Accuracy score of test KNN
97.55102040816327
[[160   4]
 [  2  79]]
Accuracy score of train KNN
96.25550660792952
Accuracy score of test KNN
97.55102040816327
[[160   4]
 [  2  79]]
Accuracy score of train KNN
96.25550660792952
Accuracy score of test KNN
97.55102040816327
[[160   4]
 [  3  78]]
Accuracy score of train KNN
96.0352422907489
Accuracy score of test KNN
97.14285714285714
```

In [12]:
```
plt.figure(figsize=(12, 6))
plt.plot(range(1, 21),knn, color='red', linestyle='dashed', marker='o', mar
plt.title('Accuracy for different  K Value')
plt.xlabel('K Value')
plt.ylabel('Accuracy')
```

Out[12]:
```
Text(0, 0.5, 'Accuracy')
```



In [13]:
```
# Fitting SVM to the Training set

from sklearn.svm import SVC
classifier = SVC(kernel = 'linear', random_state = 0)

trained_model=classifier.fit(X_train,y_train)
trained_model.fit(X_train,y_train )
```

Out[13]:
```
▾                    SVC
SVC(kernel='linear', random_state=0)
```

In [14]:
```
# Predicting the Test set results

y_pred = classifier.predict(X_test)
```

In [15]:

```
# Making the Confusion Matrix

from sklearn.metrics import confusion_matrix
cm_SVM = confusion_matrix(y_test, y_pred)
print(cm_SVM)
print("Accuracy score of train SVM")
print(accuracy_score(y_train, trained_model.predict(X_train))*100)
```

```
[[160   4]
 [  4  77]]
Accuracy score of train SVM
96.47577092511013
```

In [16]:

```
print("Accuracy score of test SVM")
print(accuracy_score(y_test, y_pred)*100)
```

```
Accuracy score of test SVM
96.73469387755102
```

In [17]:

```
from sklearn.preprocessing import StandardScaler

X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
# Example query
query_data = pd.DataFrame({

    'unif_cell_size': [2],
    'unif_cell_shape': [3]
})

# Scale the query data using the same scaler
query_data_scaled = sc.transform(query_data)

# Make predictions for the query data
query_prediction = classifier.predict(query_data_scaled)
print(f"Predicted class for query data: {query_prediction}")
```

```
Predicted class for query data: [0]

C:\Users\msuse\anaconda3\Lib\site-packages\sklearn\base.py:457: UserWarnin
g: X has feature names, but StandardScaler was fitted without feature name
s
  warnings.warn(
```

In [ ]: