Cancer-Gene-Expression-Analysis- (/github/ShubhaTiwarii/Cancer-Gene-Expression-Analysis-/tree/main)
/
Cancer Gene Expression Analysis.ipynb (/github/ShubhaTiwarii/Cancer-Gene-Expression-Analysis-/tree/main/Cancer Gene Expression Analysis.ipynb)

CANCER GENE EXPRESSION ANALYSIS USING ML

Cancer gene expression analysis using machine learning focuses on identifying patterns in gene expression data to classify cancer types, such as Acute Lymphoblastic Leukemia (ALL) and Acute Myeloid Leukemia (AML). The dataset typically includes gene accession numbers, descriptions, and expression levels for each sample. Dimensionality reduction techniques like PCA are applied to handle the high-dimensional data efficiently and capture the most relevant features. Random Forest and Neural Networks are utilized for classification, leveraging their robustness and ability to capture non-linear patterns in the data. This approach aids in accurate cancer type prediction and highlights important genes for further biological insights.

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

df_actual = pd.read_csv("actual.csv")
df_ind = pd.read_csv("data_set_ALL_AML_independent.csv")
df_train = pd.read_csv("data_set_ALL_AML_train.csv")

df_actual.head(10)
```

Out[1]:

	patient	cancer
0	1	ALL
1	2	ALL
2	3	ALL
3	4	ALL
4	5	ALL
5	6	ALL
6	7	ALL
7	8	ALL
8	9	ALL
9	10	ALL

In [2]:
```
df_ind.head(10)
```

Out[2]:

	Gene Description	Gene Accession Number	39	call	40	call.1	42	call.2	47	call.3	...	65	call.29	66	call.30	63	call.31	64	call.32
0	AFFX-BioB-5_at (endogenous control)	AFFX-BioB-5_at	-342	A	-87	A	22	A	-243	A	...	-62	A	-58	A	-161	A	-48	A
1	AFFX-BioB-M_at (endogenous control)	AFFX-BioB-M_at	-200	A	-248	A	-153	A	-218	A	...	-198	A	-217	A	-215	A	-531	A
2	AFFX-BioB-3_at (endogenous control)	AFFX-BioB-3_at	41	A	262	A	17	A	-163	A	...	-5	A	63	A	-46	A	-124	A
3	AFFX-BioC-5_at (endogenous control)	AFFX-BioC-5_at	328	A	295	A	276	A	182	A	...	141	A	95	A	146	A	431	A
4	AFFX-BioC-3_at (endogenous control)	AFFX-BioC-3_at	-224	A	-226	A	-211	A	-289	A	...	-256	A	-191	A	-172	A	-496	A
5	AFFX-BioDn-5_at (endogenous control)	AFFX-BioDn-5_at	-427	A	-493	A	-250	A	-268	A	...	-206	A	-230	A	-596	A	-696	A
6	AFFX-BioDn-3_at (endogenous control)	AFFX-BioDn-3_at	-656	A	367	A	55	A	-285	A	...	-298	A	-86	A	-122	A	-1038	A
7	AFFX-CreX-5_at (endogenous control)	AFFX-CreX-5_at	-292	A	-452	A	-141	A	-172	A	...	-218	A	-152	A	-341	A	-441	A
8	AFFX-CreX-3_at (endogenous control)	AFFX-CreX-3_at	137	A	194	A	0	A	52	A	...	-14	A	-6	A	171	A	235	A
9	AFFX-BioB-5_st (endogenous control)	AFFX-BioB-5_st	-144	A	162	A	500	A	-134	A	...	100	A	-249	A	-147	A	157	A

10 rows × 70 columns

In [3]:
```
df_train.head(10)
```

Out[3]:

	Gene Description	Gene Accession Number	1	call	2	call.1	3	call.2	4	call.3	...	29	call.33	30	call.34	31	call.35	32	call.36
0	AFFX-BioB-5_at (endogenous control)	AFFX-BioB-5_at	-214	A	-139	A	-76	A	-135	A	...	15	A	-318	A	-32	A	-124	A
1	AFFX-BioB-M_at (endogenous control)	AFFX-BioB-M_at	-153	A	-73	A	-49	A	-114	A	...	-114	A	-192	A	-49	A	-79	A
2	AFFX-BioB-3_at (endogenous control)	AFFX-BioB-3_at	-58	A	-1	A	-307	A	265	A	...	2	A	-95	A	49	A	-37	A
3	AFFX-BioC-5_at (endogenous control)	AFFX-BioC-5_at	88	A	283	A	309	A	12	A	...	193	A	312	A	230	P	330	A
4	AFFX-BioC-3_at (endogenous control)	AFFX-BioC-3_at	-295	A	-264	A	-376	A	-419	A	...	-51	A	-139	A	-367	A	-188	A
5	AFFX-BioDn-5_at (endogenous control)	AFFX-BioDn-5_at	-558	A	-400	A	-650	A	-585	A	...	-155	A	-344	A	-508	A	-423	A
6	AFFX-BioDn-3_at (endogenous control)	AFFX-BioDn-3_at	199	A	-330	A	33	A	158	A	...	29	A	324	A	-349	A	-31	A
7	AFFX-CreX-5_at (endogenous control)	AFFX-CreX-5_at	-176	A	-168	A	-367	A	-253	A	...	-105	A	-237	A	-194	A	-223	A
8	AFFX-CreX-3_at (endogenous control)	AFFX-CreX-3_at	252	A	101	A	206	A	49	A	...	42	A	105	A	34	A	-82	A
9	AFFX-BioB-5_st (endogenous control)	AFFX-BioB-5_st	206	A	74	A	-215	A	31	A	...	524	A	167	A	-56	A	176	A

10 rows × 78 columns

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

Out[4]: (72, 2)

In [5]:
```
df_ind.shape
```

Out[5]: (7129, 70)

In [6]:
```
df_train.shape
```

Out[6]: (7129, 78)

In [7]:
```
df_actual.info()
df_ind.info()
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 72 entries, 0 to 71
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype 
---  ------   --------------  ----- 
 0   patient  72 non-null     int64 
 1   cancer   72 non-null     object
dtypes: int64(1), object(1)
memory usage: 1.3+ KB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7129 entries, 0 to 7128
Data columns (total 70 columns):
 #   Column                 Non-Null Count  Dtype 
---  ------                 --------------  ----- 
 0   Gene Description       7129 non-null   object
 1   Gene Accession Number  7129 non-null   object
 2   39                     7129 non-null   int64 
 3   call                   7129 non-null   object
 4   40                     7129 non-null   int64 
 5   call.1                 7129 non-null   object
 6   42                     7129 non-null   int64 
 7   call.2                 7129 non-null   object
 8   47                     7129 non-null   int64 
 9   call.3                 7129 non-null   object
 10  48                     7129 non-null   int64 
 11  call.4                 7129 non-null   object
 12  49                     7129 non-null   int64 
 13  call.5                 7129 non-null   object
 14  41                     7129 non-null   int64 
 15  call.6                 7129 non-null   object
 16  43                     7129 non-null   int64 
 17  call.7                 7129 non-null   object
 18  44                     7129 non-null   int64 
 19  call.8                 7129 non-null   object
 20  45                     7129 non-null   int64 
 21  call.9                 7129 non-null   object
 22  46                     7129 non-null   int64 
 23  call.10                7129 non-null   object
 24  70                     7129 non-null   int64 
 25  call.11                7129 non-null   object
 26  71                     7129 non-null   int64 
 27  call.12                7129 non-null   object
 28  72                     7129 non-null   int64 
 29  call.13                7129 non-null   object
 30  68                     7129 non-null   int64 
 31  call.14                7129 non-null   object
 32  69                     7129 non-null   int64 
 33  call.15                7129 non-null   object
 34  67                     7129 non-null   int64 
 35  call.16                7129 non-null   object
 36  55                     7129 non-null   int64 
 37  call.17                7129 non-null   object
 38  56                     7129 non-null   int64 
 39  call.18                7129 non-null   object
 40  59                     7129 non-null   int64 
 41  call.19                7129 non-null   object
 42  52                     7129 non-null   int64 
 43  call.20                7129 non-null   object
 44  53                     7129 non-null   int64 
 45  call.21                7129 non-null   object
 46  51                     7129 non-null   int64 
 47  call.22                7129 non-null   object
 48  50                     7129 non-null   int64 
 49  call.23                7129 non-null   object
 50  54                     7129 non-null   int64 
 51  call.24                7129 non-null   object
 52  57                     7129 non-null   int64 
 53  call.25                7129 non-null   object
 54  58                     7129 non-null   int64 
 55  call.26                7129 non-null   object
 56  60                     7129 non-null   int64 
 57  call.27                7129 non-null   object
 58  61                     7129 non-null   int64 
```

```
 59  call.28               7129 non-null   object
 60  65                    7129 non-null   int64 
 61  call.29               7129 non-null   object
 62  66                    7129 non-null   int64 
 63  call.30               7129 non-null   object
 64  63                    7129 non-null   int64 
 65  call.31               7129 non-null   object
 66  64                    7129 non-null   int64 
 67  call.32               7129 non-null   object
 68  62                    7129 non-null   int64 
 69  call.33               7129 non-null   object
dtypes: int64(34), object(36)
memory usage: 3.8+ MB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7129 entries, 0 to 7128
Data columns (total 78 columns):
 #   Column                 Non-Null Count  Dtype 
---  ------                 --------------  ----- 
 0   Gene Description       7129 non-null   object
 1   Gene Accession Number  7129 non-null   object
 2   1                      7129 non-null   int64 
 3   call                   7129 non-null   object
 4   2                      7129 non-null   int64 
 5   call.1                 7129 non-null   object
 6   3                      7129 non-null   int64 
 7   call.2                 7129 non-null   object
 8   4                      7129 non-null   int64 
 9   call.3                 7129 non-null   object
 10  5                      7129 non-null   int64 
 11  call.4                 7129 non-null   object
 12  6                      7129 non-null   int64 
 13  call.5                 7129 non-null   object
 14  7                      7129 non-null   int64 
 15  call.6                 7129 non-null   object
 16  8                      7129 non-null   int64 
 17  call.7                 7129 non-null   object
 18  9                      7129 non-null   int64 
 19  call.8                 7129 non-null   object
 20  10                     7129 non-null   int64 
 21  call.9                 7129 non-null   object
 22  11                     7129 non-null   int64 
 23  call.10                7129 non-null   object
 24  12                     7129 non-null   int64 
 25  call.11                7129 non-null   object
 26  13                     7129 non-null   int64 
 27  call.12                7129 non-null   object
 28  14                     7129 non-null   int64 
 29  call.13                7129 non-null   object
 30  15                     7129 non-null   int64 
 31  call.14                7129 non-null   object
 32  16                     7129 non-null   int64 
 33  call.15                7129 non-null   object
 34  17                     7129 non-null   int64 
 35  call.16                7129 non-null   object
 36  18                     7129 non-null   int64 
 37  call.17                7129 non-null   object
 38  19                     7129 non-null   int64 
 39  call.18                7129 non-null   object
 40  20                     7129 non-null   int64 
 41  call.19                7129 non-null   object
 42  21                     7129 non-null   int64 
 43  call.20                7129 non-null   object
 44  22                     7129 non-null   int64 
 45  call.21                7129 non-null   object
 46  23                     7129 non-null   int64 
 47  call.22                7129 non-null   object
 48  24                     7129 non-null   int64 
 49  call.23                7129 non-null   object
 50  25                     7129 non-null   int64 
 51  call.24                7129 non-null   object
 52  26                     7129 non-null   int64 
 53  call.25                7129 non-null   object
 54  27                     7129 non-null   int64 
```

```
 55  call.26               7129 non-null   object
 56  34                    7129 non-null   int64
 57  call.27               7129 non-null   object
 58  35                    7129 non-null   int64
 59  call.28               7129 non-null   object
 60  36                    7129 non-null   int64
 61  call.29               7129 non-null   object
 62  37                    7129 non-null   int64
 63  call.30               7129 non-null   object
 64  38                    7129 non-null   int64
 65  call.31               7129 non-null   object
 66  28                    7129 non-null   int64
 67  call.32               7129 non-null   object
 68  29                    7129 non-null   int64
 69  call.33               7129 non-null   object
 70  30                    7129 non-null   int64
 71  call.34               7129 non-null   object
 72  31                    7129 non-null   int64
 73  call.35               7129 non-null   object
 74  32                    7129 non-null   int64
 75  call.36               7129 non-null   object
 76  33                    7129 non-null   int64
 77  call.37               7129 non-null   object
dtypes: int64(38), object(40)
memory usage: 4.2+ MB
```

In [8]:

```python
cols_train = [col for col in df_train if "call" in col]
cols_test = [col for col in df_ind if "call" in col]
df_train.drop(cols_train, axis=1, inplace=True)
df_ind.drop(cols_test, axis=1, inplace=True)
```

In [9]:

```python
df_train = df_train.T
df_ind = df_ind.T
```

In [10]:

```python
df_train.columns = df_train.iloc[1].values
df_train.drop(["Gene Description", "Gene Accession Number"], axis=0, inplace=True)
df_ind.columns = df_ind.iloc[1].values
df_ind.drop(["Gene Description", "Gene Accession Number"], axis=0, inplace=True)
```

In [11]:

```python
df_train.head()
```

Out[11]:

	AFFX-BioB-5_at	AFFX-BioB-M_at	AFFX-BioB-3_at	AFFX-BioC-5_at	AFFX-BioC-3_at	AFFX-BioDn-5_at	AFFX-BioDn-3_at	AFFX-CreX-5_at	AFFX-CreX-3_at	AFFX-BioB-5_st	...	U48730_at	U58516_at	U73738_at	X06956_at	X166
1	-214	-153	-58	88	-295	-558	199	-176	252	206	...	185	511	-125	389	
2	-139	-73	-1	283	-264	-400	-330	-168	101	74	...	169	837	-36	442	
3	-76	-49	-307	309	-376	-650	33	-367	206	-215	...	315	1199	33	168	
4	-135	-114	265	12	-419	-585	158	-253	49	31	...	240	835	218	174	
5	-106	-125	-76	168	-230	-284	4	-122	70	252	...	156	649	57	504	

5 rows × 7129 columns

In [12]:

```python
df_ind.head()
```

Out[12]:

	AFFX-BioB-5_at	AFFX-BioB-M_at	AFFX-BioB-3_at	AFFX-BioC-5_at	AFFX-BioC-3_at	AFFX-BioDn-5_at	AFFX-BioDn-3_at	AFFX-CreX-5_at	AFFX-CreX-3_at	AFFX-BioB-5_st	...	U48730_at	U58516_at	U73738_at	X06956_at	X16(
39	-342	-200	41	328	-224	-427	-656	-292	137	-144	...	277	1023	67	214	
40	-87	-248	262	295	-226	-493	367	-452	194	162	...	83	529	-295	352	
42	22	-153	17	276	-211	-250	55	-141	0	500	...	413	399	16	558	
47	-243	-218	-163	182	-289	-268	-285	-172	52	-134	...	174	277	6	81	
48	-130	-177	-28	266	-170	-326	-222	-93	10	159	...	233	643	51	450	

5 rows × 7129 columns

In [13]:
```
df_train["patient"] = df_train.index.values
df_ind["patient"] = df_ind.index.values
```

In [14]:
```
df_train = df_train.astype("int32")
df_ind = df_ind.astype("int32")
```

In [15]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df_actual["cancer"] = le.fit_transform(df_actual["cancer"])
df_actual.head()
```

Out[15]:

	patient	cancer
0	1	0
1	2	0
2	3	0
3	4	0
4	5	0

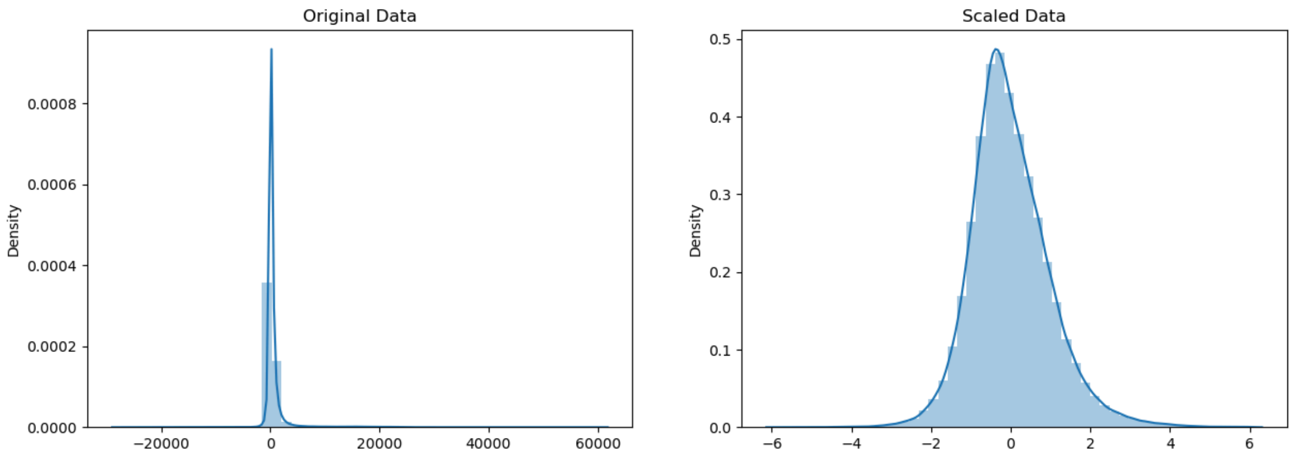
In [16]:
```
df_train = pd.merge(df_train, df_actual, on="patient")
df_ind = pd.merge(df_ind, df_actual, on="patient")
```

In [17]:
```
X_train = df_train.drop(columns=["patient", "cancer"])
y_train = df_train["cancer"]
X_test = df_ind.drop(columns=["patient", "cancer"])
y_test = df_ind["cancer"]
```

In [18]:
```
from sklearn.preprocessing import StandardScaler
scale = StandardScaler()
X_train_s = scale.fit_transform(X_train)
X_test_s = scale.transform(X_test)
```

In [19]:
```
fig, ax = plt.subplots(ncols=2, figsize=(15,5))
sns.distplot(np.concatenate(X_train.values), ax=ax[0]).set_title('Original Data')
sns.distplot(np.concatenate(X_train_s), ax=ax[1]).set_title('Scaled Data')
plt.tight_layout
plt.show()
```

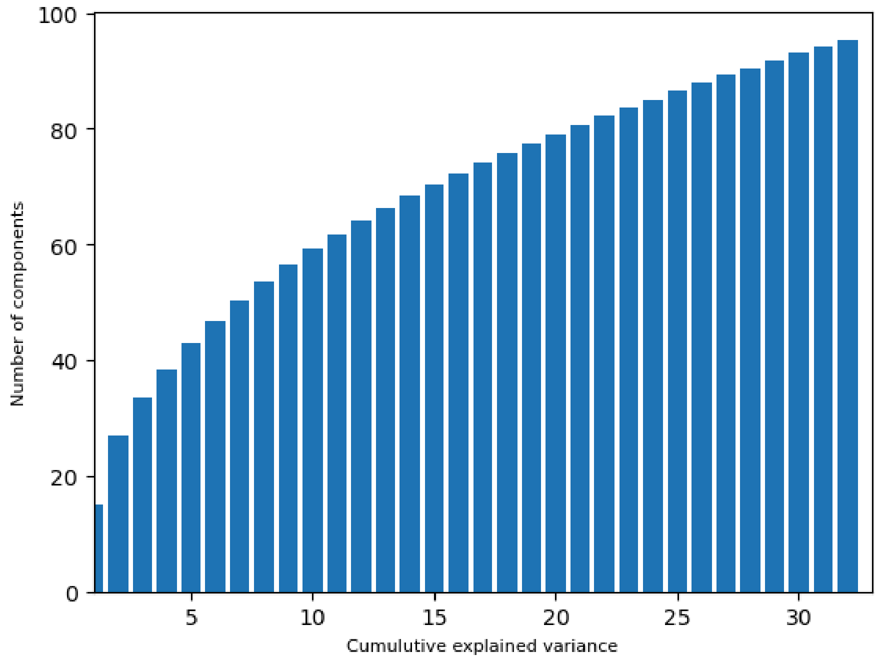
```
C:\Users\shubh\AppData\Local\Temp\ipykernel_19248\1189716076.py:2: UserWarning: 

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(np.concatenate(X_train.values), ax=ax[0]).set_title('Original Data')
C:\Users\shubh\AppData\Local\Temp\ipykernel_19248\1189716076.py:3: UserWarning: 

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(np.concatenate(X_train_s), ax=ax[1]).set_title('Scaled Data')
```



Principal Component Analysis

In [20]:
```python
from sklearn.decomposition import PCA

pca = PCA()
pca.fit_transform(X_train_s)
total = sum(pca.explained_variance_)
k = 0
cur_var = 0
while cur_var / total < 0.95:
    cur_var += pca.explained_variance_[k]
    k = k + 1
print(" Around 95% of the variance explained by ",k," features", sep='')

pca = PCA(n_components=k)
X_train_p = pca.fit(X_train_s)
X_train_p = pca.transform(X_train_s)
X_test_p = pca.transform(X_test_s)

exp_var = pca.explained_variance_ratio_.cumsum()
exp_var = exp_var*100
plt.bar(range(1, k + 1), exp_var)
plt.xlabel("Cumulutive explained variance", fontsize=8)
plt.ylabel("Number of components", fontsize=8)
plt.xlim((1, k + 1))
plt.show()
```

```
 Around 95% of the variance explained by 32 features
```

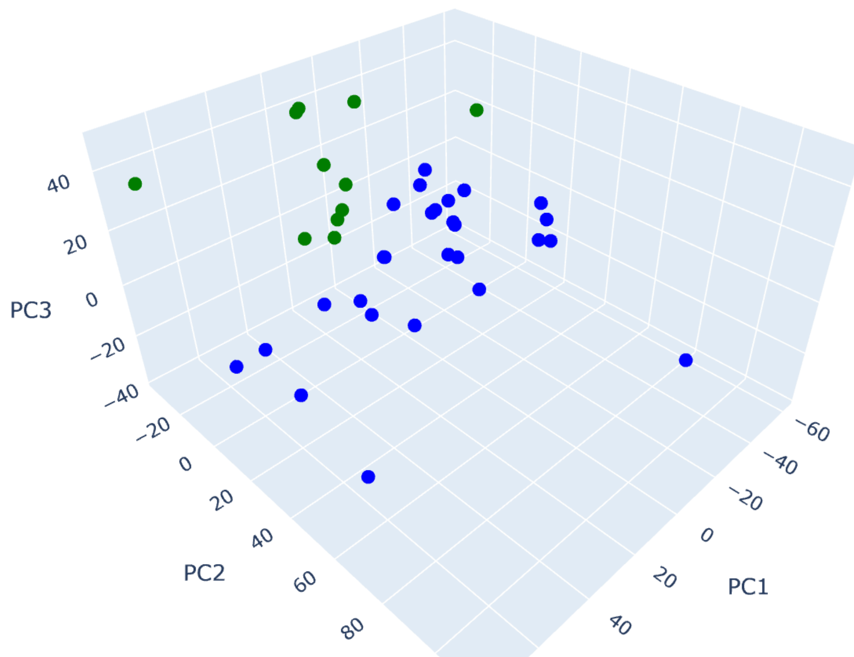


In [21]:
```
print(X_train_p.shape)
print(X_test_p.shape)
```

```
(38, 32)
(34, 32)
```

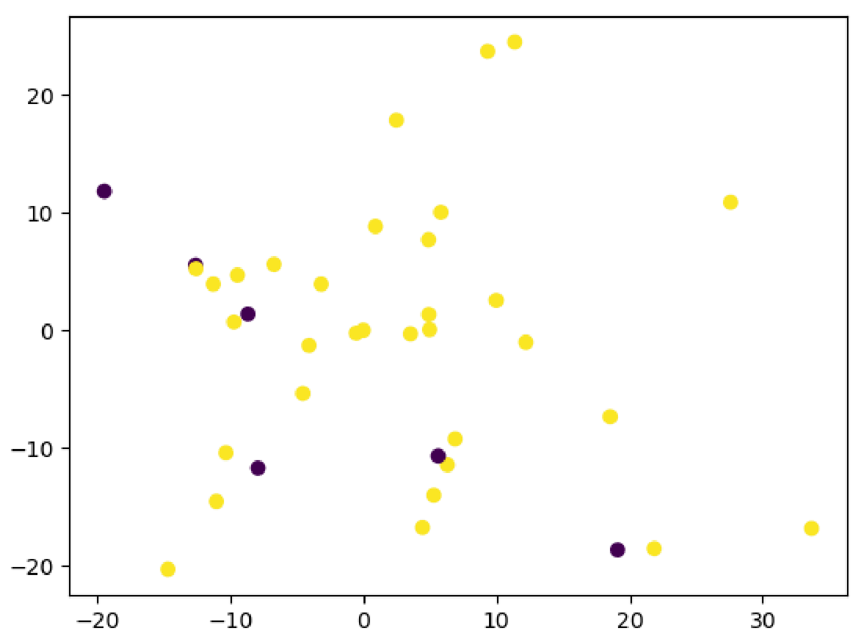
In [22]:
```
import plotly.express as px
import plotly.graph_objects as go

# PCA to reduce to 3 components
pca3 = PCA(n_components=3).fit_transform(X_train_s)

colr = ['blue' if label == 0 else 'green' for label in y_train]

fig = go.Figure(data=[go.Scatter3d(
    x=pca3[:, 0],
    y=pca3[:, 1],
    z=pca3[:, 2],
    mode='markers',
    marker=dict(
        size=5,
        color=colr,
    )
)])

fig.update_layout(
    title="Top 3 PCA components",
    scene=dict(
        xaxis_title="PC1",
        yaxis_title="PC2",
        zaxis_title="PC3"
    ),
    margin=dict(l=0, r=0, b=0, t=40)
)
fig.show()
```

Top 3 PCA components



In [23]:
```
from sklearn.cluster import KMeans
from scipy.cluster.hierarchy import dendrogram, linkage, fcluster

km = KMeans(n_clusters=2, n_init=20)
km.fit(df_train)

X_train_red = PCA().fit_transform(X_train_s)
plt.scatter(x=X_train_red[0], y=X_train_red[1], c=km.labels_, cmap='viridis')
plt.show()
```

```
C:\Users\shubh\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1446: UserWarning:

KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You c
an avoid it by setting the environment variable OMP_NUM_THREADS=1.
```



Neural Network for Gene Expression Classification:

In [24]:
```
from tensorflow.random import set_seed
set_seed(0)
from tensorflow import keras
from tensorflow.keras import layers
from sklearn.metrics import accuracy_score, confusion_matrix

NN = keras.Sequential([
    layers.Dense(32, activation='relu', input_shape=X_train_p[1].shape),
    layers.Dense(16, activation='relu'),
    layers.Dense(1, activation='sigmoid')
])

NN.compile(
    loss='binary_crossentropy',
    optimizer='adam',
    metrics=['binary_accuracy']
)

es = keras.callbacks.EarlyStopping(
    patience=5,
    min_delta=0.005,
    restore_best_weights=True,
)
```

```
C:\Users\shubh\anaconda3\Lib\site-packages\keras\src\layers\core\dense.py:87: UserWarning:

Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input
(shape)` object as the first layer in the model instead.
```

In [25]:
```
history = NN.fit(
    X_train_p, y_train,
    validation_data=(X_test_p, y_test),
    batch_size = 8,
    epochs = 200,
    callbacks=[es]
)
```

```
Epoch 1/200
5/5 ━━━━━━━━━━━━━━━━━━━━ 2s 100ms/step - binary_accuracy: 0.3715 - loss: 4.4679 - val_binary_accuracy: 0.6176 - val
_loss: 1.1435
Epoch 2/200
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step - binary_accuracy: 0.4029 - loss: 3.1870 - val_binary_accuracy: 0.6471 - val_
loss: 1.0548
Epoch 3/200
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - binary_accuracy: 0.5301 - loss: 2.4374 - val_binary_accuracy: 0.6765 - val_
loss: 0.9747
Epoch 4/200
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - binary_accuracy: 0.6432 - loss: 1.8097 - val_binary_accuracy: 0.7353 - val_
loss: 0.9167
Epoch 5/200
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - binary_accuracy: 0.7267 - loss: 1.3022 - val_binary_accuracy: 0.7059 - val_
loss: 0.8756
Epoch 6/200
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step - binary_accuracy: 0.7825 - loss: 0.9240 - val_binary_accuracy: 0.7353 - val_
loss: 0.8522
Epoch 7/200
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - binary_accuracy: 0.8869 - loss: 0.6458 - val_binary_accuracy: 0.7353 - val_
loss: 0.8473
Epoch 8/200
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - binary_accuracy: 0.8957 - loss: 0.4265 - val_binary_accuracy: 0.7353 - val_
loss: 0.8549
Epoch 9/200
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - binary_accuracy: 0.8957 - loss: 0.2496 - val_binary_accuracy: 0.7647 - val_
loss: 0.8660
Epoch 10/200
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step - binary_accuracy: 0.9478 - loss: 0.1517 - val_binary_accuracy: 0.7647 - val_
loss: 0.8754
Epoch 11/200
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - binary_accuracy: 0.9478 - loss: 0.1053 - val_binary_accuracy: 0.7647 - val_
loss: 0.8812
```

In [26]:

```python
from sklearn.metrics import accuracy_score

pred_prob = NN.predict(X_test_p)

NNpred = np.argmax(pred_prob, axis=1)
print('Accuracy for Neural Network ', round(accuracy_score(y_test, NNpred), 3))
```

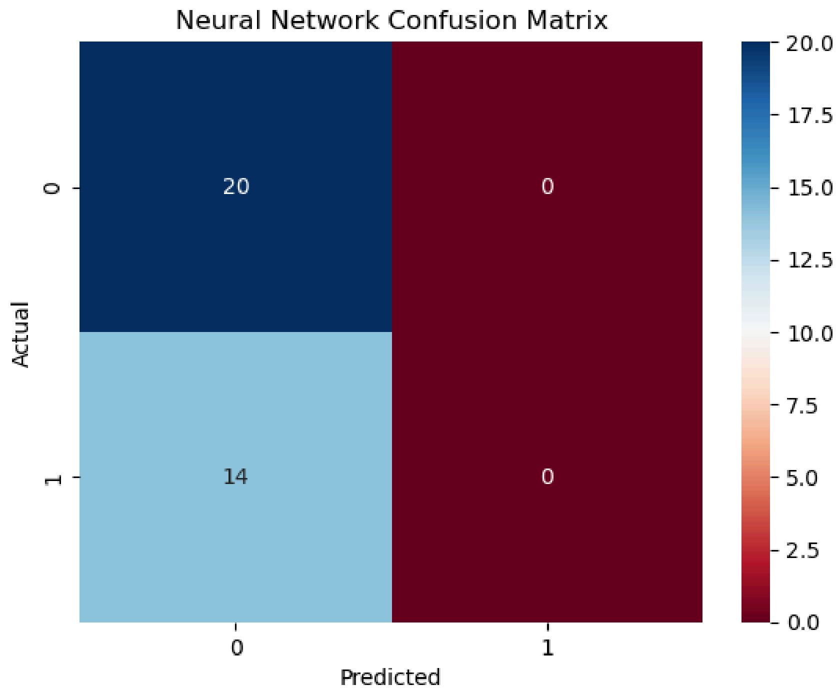
```
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 78ms/step
Accuracy for Neural Network  0.588
```

In [27]:

```python
NN_conf = confusion_matrix(y_test, NNpred)

ax = plt.subplot()
sns.heatmap(NN_conf, annot=True, ax = ax, fmt='g', cmap='RdBu')

ax.set_xlabel('Predicted')
ax.set_ylabel('Actual')
ax.set_title('Neural Network Confusion Matrix')
```

Out[27]:

```
Text(0.5, 1.0, 'Neural Network Confusion Matrix')
```



Random Forest:

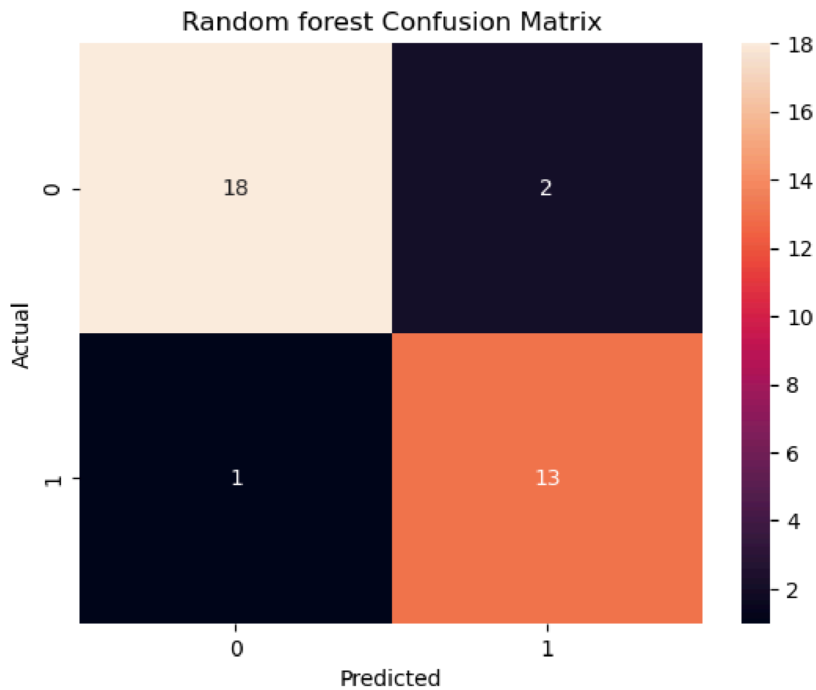
In [28]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import RandomizedSearchCV
from sklearn import metrics

RFC_parameters = {
    "bootstrap": [True, False],
    "n_estimators": np.arange(60, 101, 10),
    "max_features": [0.6, 0.7, 0.8],
    "min_samples_leaf": [8, 10, 12],
    "min_samples_split": [3, 5, 7]
}

RFC_search = RandomizedSearchCV(
    estimator=RandomForestClassifier(),
    param_distributions=RFC_parameters,
    n_iter=50,
    scoring="f1",
    n_jobs=-1,
    verbose=2,
    cv=3,
    random_state=42
)

RFC_search.fit(X_train_s, y_train)

RFC_best_est = RFC_search.best_estimator_

RFC_pred = RFC_best_est.predict(X_test_s)

f1_score = metrics.f1_score(y_test, RFC_pred)
print('f1-score of RandomForest Classifier is', f1_score)
print("Classification scores :", metrics.classification_report(y_test, RFC_pred))
```

```
Fitting 3 folds for each of 50 candidates, totalling 150 fits
f1-score of RandomForest Classifier is 0.896551724137931
Classification scores :               precision    recall  f1-score   support

           0       0.95      0.90      0.92        20
           1       0.87      0.93      0.90        14

    accuracy                           0.91        34
   macro avg       0.91      0.91      0.91        34
weighted avg       0.91      0.91      0.91        34
```

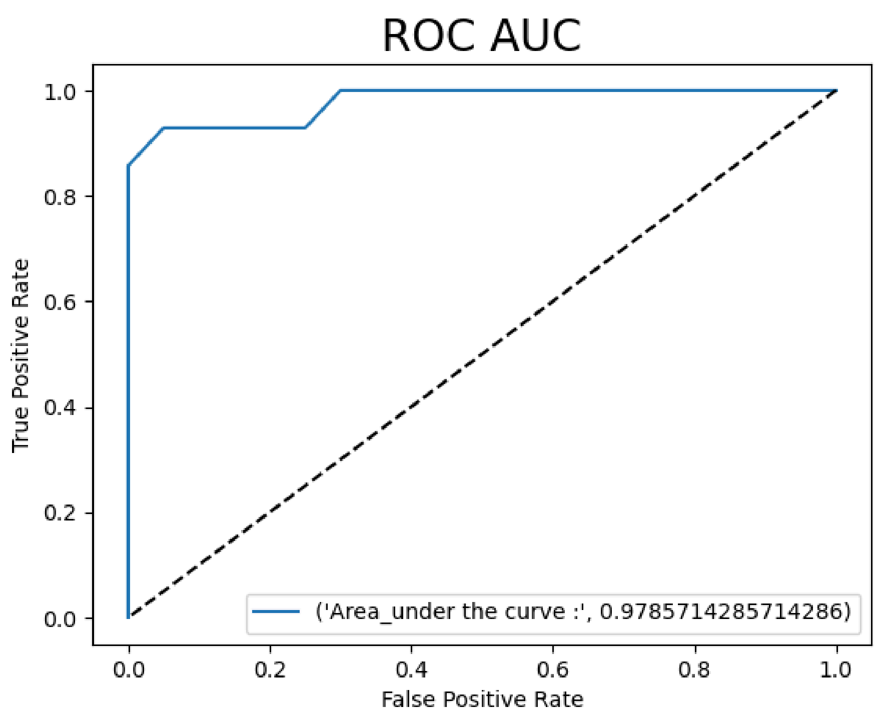
In [29]:

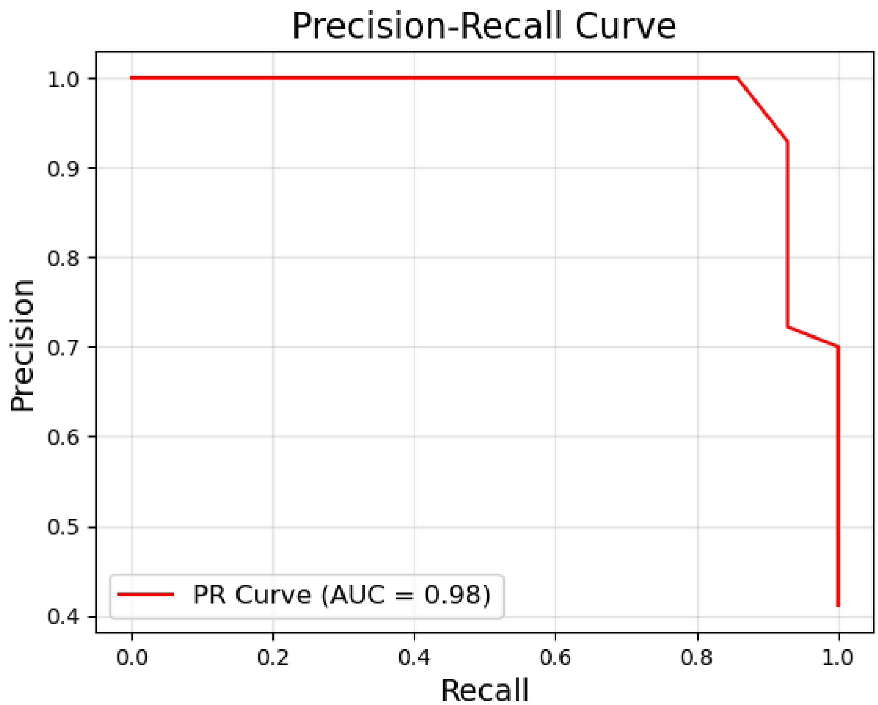
```
ax = plt.subplot()
sns.heatmap(metrics.confusion_matrix(y_test, RFC_pred), annot=True, ax=ax)
plt.title("Random forest Confusion Matrix")
ax.set_xlabel('Predicted')
ax.set_ylabel('Actual')
```

Out[29]: `Text(50.722222222222214, 0.5, 'Actual')`



In [30]:
```
from sklearn.metrics import precision_score, recall_score, f1_score, roc_auc_score, precision_recall_curve, auc, ro

# ROC curve
RFC_predicted_prob = RFC_best_est.predict_proba(X_test_s)[:, 1]
fpr, tpr, thresholds = metrics.roc_curve(y_test, RFC_predicted_prob)
plt.subplot()
plt.plot(fpr, tpr, label = ("Area_under the curve :", metrics.auc(fpr, tpr)))
plt.plot([1,0], [1,0], linestyle = "dashed", color ="k")
plt.legend(loc = "best")
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title("ROC AUC",fontsize=20)
plt.show()

precision, recall, thresholds = precision_recall_curve(y_test, RFC_predicted_prob)

# Area Under the Curve (AUC) for Precision-Recall
pr_auc = auc(recall, precision)

# Precision-Recall curve
plt.plot(recall, precision, label=f"PR Curve (AUC = {pr_auc:.2f})", color='r')
plt.title("Precision-Recall Curve", fontsize=16)
plt.xlabel("Recall", fontsize=14)
plt.ylabel("Precision", fontsize=14)
plt.legend(loc="best", fontsize=12)
plt.grid(alpha=0.3)
plt.show()
```

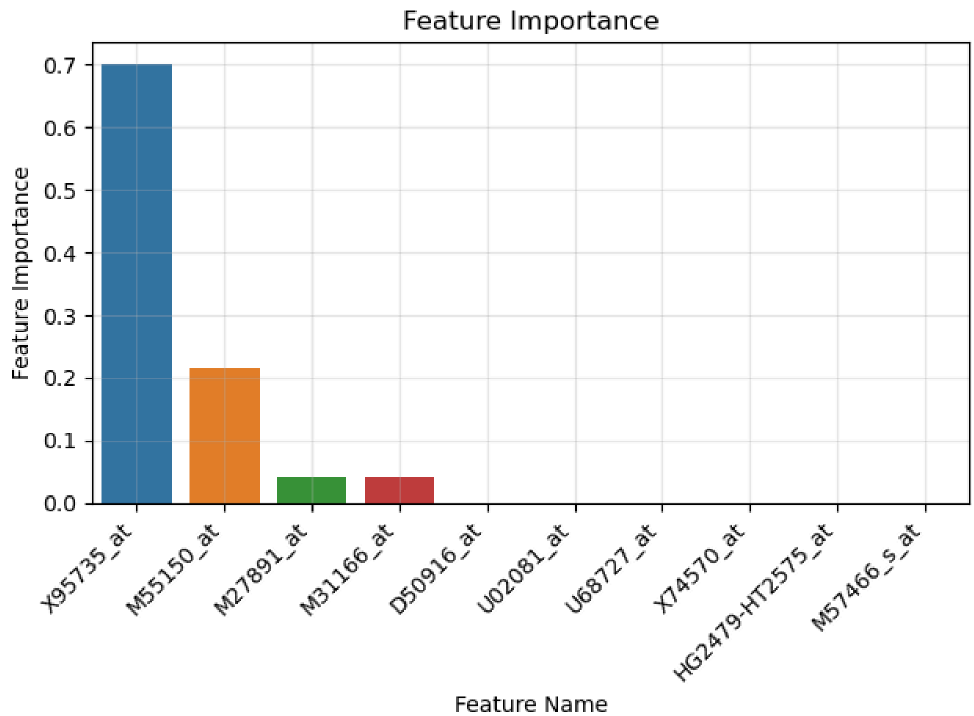




In [31]:

```
# number of features with zero importance
zero_importance_count = (RFC_best_est.feature_importances_ == 0).sum()
print(f"Quantity of features with 0 importance: {zero_importance_count}")

# features with non-zero importance
non_zero_indices = RFC_best_est.feature_importances_ != 0
importances = RFC_best_est.feature_importances_[non_zero_indices]
feature_names = df_train.columns[:7129][non_zero_indices]  # Assuming first 7129 columns are used

importance_data = pd.DataFrame({
    'Feature': feature_names,
    'Importance': importances
}).sort_values(by='Importance', ascending=False)

print("Feature Importances:")
print(importance_data.to_string(index=False))

sns.barplot(x='Feature', y='Importance', data=importance_data)
plt.title("Feature Importance")
plt.ylabel("Feature Importance")
plt.xlabel("Feature Name")
plt.xticks(rotation=45, ha="right")
plt.grid(alpha=0.3)
plt.tight_layout()
plt.show()
```

```
Quantity of features with 0 importance: 7119
Feature Importances:
         Feature  Importance
       X95735_at    0.700000
       M55150_at    0.214286
       M27891_at    0.042459
       M31166_at    0.042459
       D50916_at    0.000133
       U02081_at    0.000133
       U68727_at    0.000133
       X74570_at    0.000133
HG2479-HT2575_at    0.000133
     M57466_s_at    0.000133
```

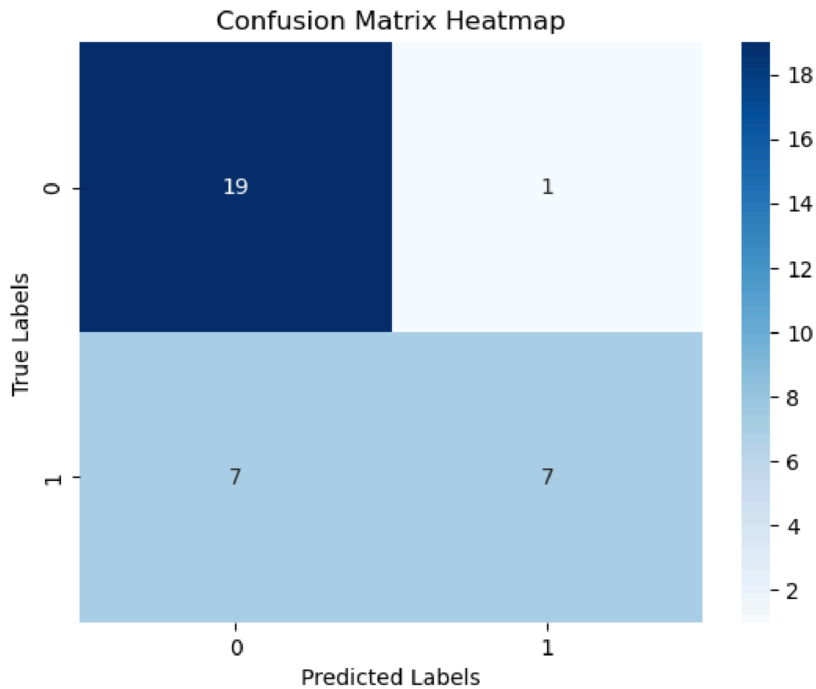


K Nearest Neighbors:

In [ ]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import RandomizedSearchCV

knn_param_grid = {
    "n_neighbors": list(range(1, 30, 5)),
    "weights": ["uniform", "distance"],
    "algorithm": ["kd_tree"],
    "leaf_size": [1, 10, 20, 30],
    "p": [1, 2]
}

random_search = RandomizedSearchCV(
    estimator=KNeighborsClassifier(),
    param_distributions=knn_param_grid,
    scoring="f1",
    n_jobs=-1,
    verbose=1
)

y_pred_knn = random_search.fit(X_train_s, y_train)

best_knn = random_search.best_estimator_
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
```

In [ ]:
```
y_test_pred_knn = best_knn.predict(X_test_s)

conf_matrix_knn = confusion_matrix(y_test, y_test_pred_knn)

sns.heatmap(conf_matrix_knn, annot=True, fmt="d", cmap="Blues", xticklabels=True, yticklabels=True)
plt.title("Confusion Matrix Heatmap")
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.show()

class_report_knn = metrics.classification_report(y_test, y_test_pred_knn)

print("\nClassification Report:\n", class_report_knn)
```




```
Classification Report:
              precision    recall  f1-score   support

           0       0.73      0.95      0.83        20
           1       0.88      0.50      0.64        14

    accuracy                           0.76        34
   macro avg       0.80      0.72      0.73        34
weighted avg       0.79      0.76      0.75        34
```

Logistic Regression:

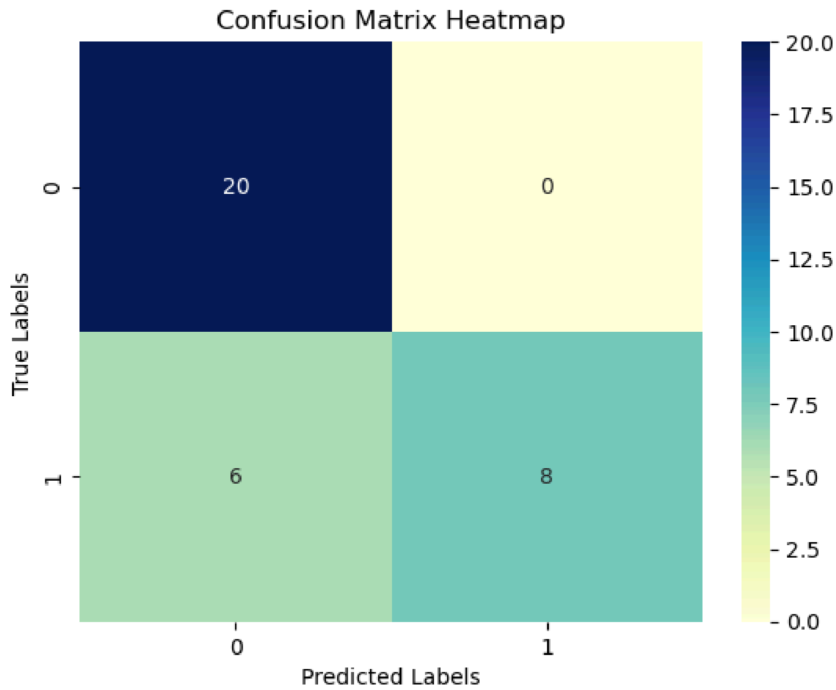
In [ ]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV

# Define the hyperparameter grid
lr_param_grid = {
    "C": [0.001, 0.01, 0.1, 1, 10],
    "penalty": ["l1", "l2"]
}


grid_search_lr = GridSearchCV(
    estimator=LogisticRegression(),
    param_grid=lr_param_grid,
    scoring="f1"
)

grid_search_lr.fit(X_train_s, y_train)

best_lr = grid_search_lr.best_estimator_
```

```
C:\Users\shubh\anaconda3\Lib\site-packages\sklearn\model_selection\_validation.py:547: FitFailedWarning:


25 fits failed out of a total of 50.
The score on these train-test partitions for these parameters will be set to nan.
If these failures are not expected, you can try to debug them by setting error_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
25 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Users\shubh\anaconda3\Lib\site-packages\sklearn\model_selection\_validation.py", line 895, in _fit_and_s
core
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\shubh\anaconda3\Lib\site-packages\sklearn\base.py", line 1474, in wrapper
    return fit_method(estimator, *args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\shubh\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.py", line 1172, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\shubh\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.py", line 67, in _check_solver
    raise ValueError(
ValueError: Solver lbfgs supports only 'l2' or None penalties, got l1 penalty.


C:\Users\shubh\anaconda3\Lib\site-packages\sklearn\model_selection\_search.py:1051: UserWarning:

One or more of the test scores are non-finite: [       nan 0.46666667        nan 0.53333333        nan 0.83333333
        nan 0.83333333        nan 0.62222222]
```

In [ ]:

```
y_test_pred_lr = best_lr.predict(X_test_s)

class_report_lr = metrics.classification_report(y_test, y_test_pred_lr)
print("Classification Report:\n", class_report_lr)

conf_matrix_lr = confusion_matrix(y_test, y_test_pred_lr)
sns.heatmap(conf_matrix_lr, annot=True, fmt="d", cmap="YlGnBu", xticklabels=True, yticklabels=True)
plt.title("Confusion Matrix Heatmap")
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.show()
```

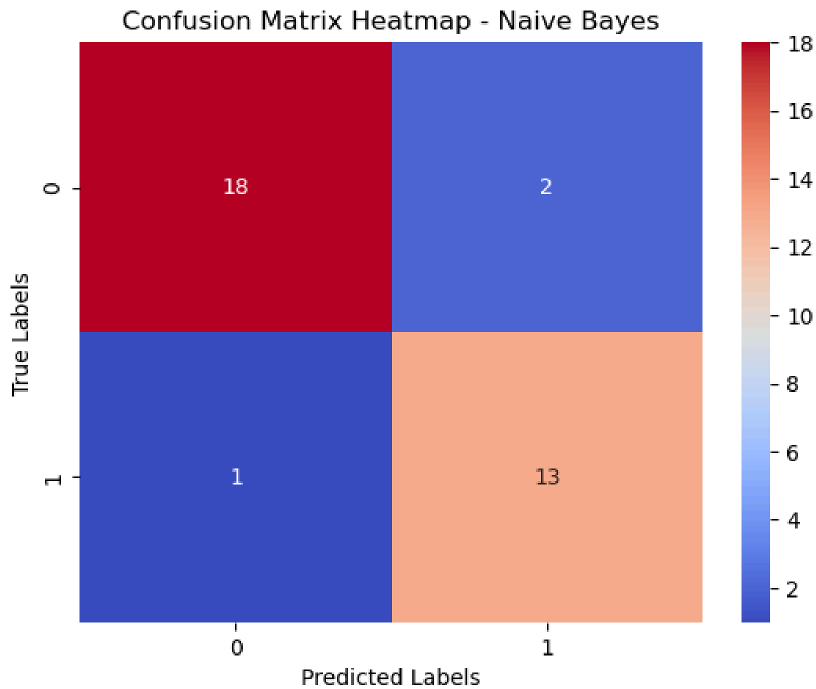
```
Classification Report:
               precision    recall  f1-score   support

           0       0.77      1.00      0.87        20
           1       1.00      0.57      0.73        14

    accuracy                           0.82        34
   macro avg       0.88      0.79      0.80        34
weighted avg       0.86      0.82      0.81        34
```



Naive Bayes:

In [ ]:
```
from sklearn.naive_bayes import GaussianNB

nb = GaussianNB()
nb.fit(X_train_s, y_train)

y_test_pred_nb = nb.predict(X_test_s)

class_report_nb = metrics.classification_report(y_test, y_test_pred_nb)
print("Classification Report:\n", class_report_nb)


conf_matrix_nb = confusion_matrix(y_test, y_test_pred_nb)
sns.heatmap(conf_matrix_nb, annot=True, fmt="d", cmap="coolwarm", xticklabels=True, yticklabels=True)
plt.title("Confusion Matrix Heatmap - Naive Bayes")
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.show()
```

```
Classification Report:
               precision    recall  f1-score   support

           0       0.95      0.90      0.92        20
           1       0.87      0.93      0.90        14

    accuracy                           0.91        34
   macro avg       0.91      0.91      0.91        34
weighted avg       0.91      0.91      0.91        34
```



Ensemble Model:

In [44]:
```
from sklearn.metrics import f1_score
import numpy as np

# Calculate F1 scores for each model
rf_f1_score = f1_score(y_test, RFC_best_est.predict(X_test_s))
knn_f1_score = f1_score(y_test, best_knn.predict(X_test_s))
nb_f1_score = f1_score(y_test, nb.predict(X_test_s))
lr_f1_score = f1_score(y_test, best_lr.predict(X_test_s))

# Compute the mean F1 score across models
mean_f1_score = np.mean([rf_f1_score, knn_f1_score, nb_f1_score, lr_f1_score])

# Calculate the weight of each model's F1 score relative to the mean
weights = {
    "rf": rf_f1_score / mean_f1_score,
    "knn": knn_f1_score / mean_f1_score,
    "nb": nb_f1_score / mean_f1_score,
    "lr": lr_f1_score / mean_f1_score
}

# Extract individual weights if needed
weight_rf = weights["rf"]
weight_knn = weights["knn"]
weight_nb = weights["nb"]
weight_lr = weights["lr"]

print(weight_rf, weight_knn, weight_nb, weight_lr )
```

```
1.1360476663356505 0.8063555114200597 1.1360476663356505 0.9215491559086396
```

In [45]:
```
from sklearn.ensemble import VotingClassifier
ensemble = VotingClassifier(estimators=[("rf", RFC_best_est), ("knn", best_knn), ("nb", nb), ("lr", best_lr)],
                                        voting="soft", weights=[weight_rf, weight_knn, weight_nb, weight_lr])
ensemble.fit(X_train_s, y_train)
```

Out[45]:

▸ **VotingClassifier**

rf	knn	nb	lr
▸ RandomForestClassifier	▸ KNeighborsClassifier	▸ GaussianNB	▸ LogisticRegression

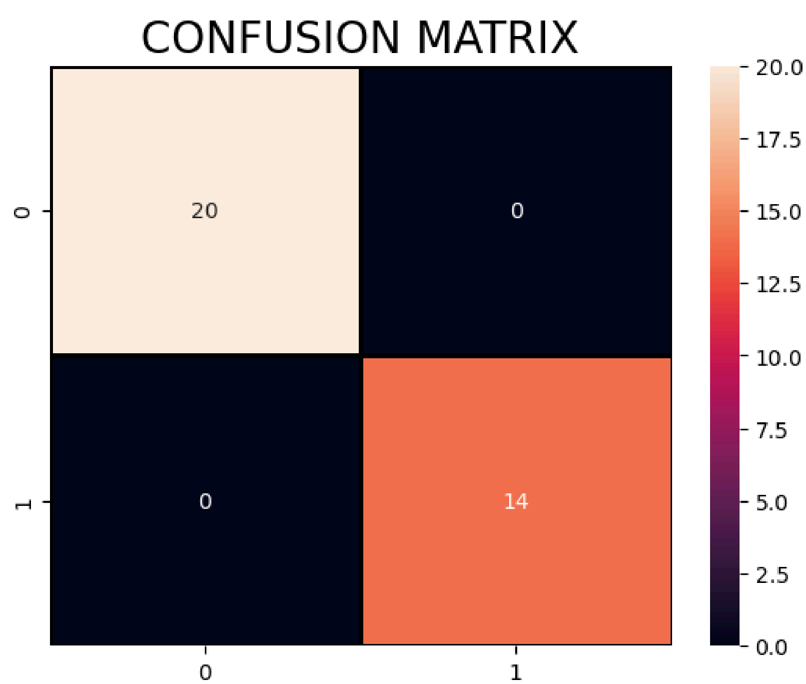
(https://scikit-learn.org/1.4/modules/generated/sklearn.ensemble.RandomForestClassifier.html) (https://scikit-learn.org/1.4/modules/generated/sklearn.neighbors.KNeighborsClas

In [48]:

```
ens_prediction = ensemble.predict(X_test_s)
f1_score = metrics.f1_score(y_test, ens_prediction)
print('Validation f1-score of Ensemble Classifier is', f1_score)
print ("\nClassification report :\n", metrics.classification_report(y_test, ens_prediction))

# Confusion matrix
sns.heatmap(metrics.confusion_matrix(y_test, ens_prediction), annot=True, fmt = "d", linecolor="k", linewidths=1)
plt.title("CONFUSION MATRIX", fontsize=20)
```

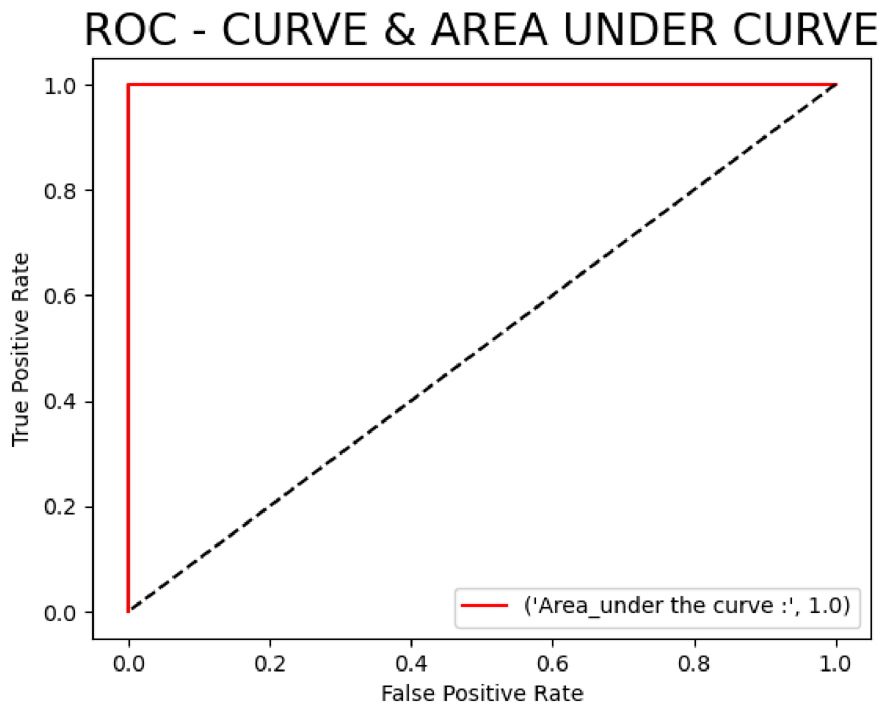
```
Validation f1-score of Ensemble Classifier is 1.0

Classification report :
               precision    recall  f1-score   support

           0       1.00      1.00      1.00        20
           1       1.00      1.00      1.00        14

    accuracy                           1.00        34
   macro avg       1.00      1.00      1.00        34
weighted avg       1.00      1.00      1.00        34
```

Out[48]: Text(0.5, 1.0, 'CONFUSION MATRIX')



In [49]:

```
# ROC curve
ens_predicted_probs = ensemble.predict_proba(X_test_s)[:, 1]
fpr, tpr, thresholds = metrics.roc_curve(y_test, ens_predicted_probs)
plt.plot(fpr, tpr, label = ("Area_under the curve :", metrics.auc(fpr, tpr)), color = "r")
plt.plot([1,0], [1,0], linestyle = "dashed", color ="k")
plt.legend(loc = "best")
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title("ROC - CURVE & AREA UNDER CURVE",fontsize=20)
plt.show()
```



Results:

The genes X95735_at, M55150_at, M27891_at, and M31166_at exhibit high feature importance, indicating their significant role in distinguishing between ALL and AML cancer types in the analysis.

Different models such as random forest , knearest neighbours, neural network, logistic regression and ensemble based models have been used and their results are found out.

The best results were shown by the ensemble model with a score 1.0.