Imports and Data Loading

In [1]:

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler

# Load datasets
cancer_df = pd.read_csv('cancer.csv')
bankrupt_df = pd.read_csv('bankrupt.csv')

# Show column names and info
print("Cancer dataset columns:\n", cancer_df.columns)
print("Bankruptcy dataset columns:\n", bankrupt_df.columns)
```

```
Cancer dataset columns:
 Index(['Patient_ID', 'Age', 'Gender', 'Country_Region', 'Year', 'Genetic_Ri
sk',
       'Air_Pollution', 'Alcohol_Use', 'Smoking', 'Obesity_Level',
       'Cancer_Type', 'Cancer_Stage', 'Treatment_Cost_USD', 'Survival_Year
s',
       'Target_Severity_Score'],
      dtype='object')
Bankruptcy dataset columns:
 Index(['Bankrupt?', ' ROA(C) before interest and depreciation before intere
st',
       ' ROA(A) before interest and % after tax',
       ' ROA(B) before interest and depreciation after tax',
       ' Operating Gross Margin', ' Realized Sales Gross Margin',
       ' Operating Profit Rate', ' Pre-tax net Interest Rate',
       ' After-tax net Interest Rate',
       ' Non-industry income and expenditure/revenue',
       ' Continuous interest rate (after tax)', ' Operating Expense Rate',
       ' Research and development expense rate', ' Cash flow rate',
       ' Interest-bearing debt interest rate', ' Tax rate (A)',
       ' Net Value Per Share (B)', ' Net Value Per Share (A)',
       ' Net Value Per Share (C)', ' Persistent EPS in the Last Four Season
s',
       ' Cash Flow Per Share', ' Revenue Per Share (Yuan ¥)',
       ' Operating Profit Per Share (Yuan ¥)',
       ' Per Share Net profit before tax (Yuan ¥)',
       ' Realized Sales Gross Profit Growth Rate',
       ' Operating Profit Growth Rate', ' After-tax Net Profit Growth Rate',
       ' Regular Net Profit Growth Rate', ' Continuous Net Profit Growth Rat
e',
       ' Total Asset Growth Rate', ' Net Value Growth Rate',
       ' Total Asset Return Growth Rate Ratio', ' Cash Reinvestment %',
       ' Current Ratio', ' Quick Ratio', ' Interest Expense Ratio',
       ' Total debt/Total net worth', ' Debt ratio %', ' Net worth/Assets',
       ' Long-term fund suitability ratio (A)', ' Borrowing dependency',
       ' Contingent liabilities/Net worth',
       ' Operating profit/Paid-in capital',
       ' Net profit before tax/Paid-in capital',
       ' Inventory and accounts receivable/Net value', ' Total Asset Turnove
r',
       ' Accounts Receivable Turnover', ' Average Collection Days',
       ' Inventory Turnover Rate (times)', ' Fixed Assets Turnover Frequenc
y',
       ' Net Worth Turnover Rate (times)', ' Revenue per person',
       ' Operating profit per person', ' Allocation rate per person',
       ' Working Capital to Total Assets', ' Quick Assets/Total Assets',
       ' Current Assets/Total Assets', ' Cash/Total Assets',
       ' Quick Assets/Current Liability', ' Cash/Current Liability',
       ' Current Liability to Assets', ' Operating Funds to Liability',
       ' Inventory/Working Capital', ' Inventory/Current Liability',
       ' Current Liabilities/Liability', ' Working Capital/Equity',
       ' Current Liabilities/Equity', ' Long-term Liability to Current Asset
s',
       ' Retained Earnings to Total Assets', ' Total income/Total expense',
       ' Total expense/Assets', ' Current Asset Turnover Rate',
       ' Quick Asset Turnover Rate', ' Working capitcal Turnover Rate',
```

```
       ' Cash Turnover Rate', ' Cash Flow to Sales', ' Fixed Assets to Asset
s',
       ' Current Liability to Liability', ' Current Liability to Equity',
       ' Equity to Long-term Liability', ' Cash Flow to Total Assets',
       ' Cash Flow to Liability', ' CFO to Assets', ' Cash Flow to Equity',
       ' Current Liability to Current Assets', ' Liability-Assets Flag',
       ' Net Income to Total Assets', ' Total assets to GNP price',
       ' No-credit Interval', ' Gross Profit to Sales',
       ' Net Income to Stockholder's Equity', ' Liability to Equity',
       ' Degree of Financial Leverage (DFL)',
       ' Interest Coverage Ratio (Interest expense to EBIT)',
       ' Net Income Flag', ' Equity to Liability'],
      dtype='object')
```

Preprocessing (Drop NAs, Keep Numeric, Scale)

In [2]:
```
# Drop rows with missing values
cancer_df_clean = cancer_df.dropna()
bankrupt_df_clean = bankrupt_df.dropna()

# Select only numerical columns
cancer_X = cancer_df_clean.select_dtypes(include=[np.number])
bankrupt_X = bankrupt_df_clean.select_dtypes(include=[np.number])

# Standardize features
scaler_cancer = StandardScaler()
cancer_X_scaled = scaler_cancer.fit_transform(cancer_X)

scaler_bankrupt = StandardScaler()
bankrupt_X_scaled = scaler_bankrupt.fit_transform(bankrupt_X)
```

In [3]:
```
from sklearn.cluster import KMeans
from sklearn.mixture import GaussianMixture
from sklearn.metrics import silhouette_score

# You can change n_clusters if you wish (try 2-5)
n_clusters = 3

# K-Means
kmeans_cancer = KMeans(n_clusters=n_clusters, random_state=42)
kmeans_cancer_labels = kmeans_cancer.fit_predict(cancer_X_scaled)

kmeans_bankrupt = KMeans(n_clusters=n_clusters, random_state=42)
kmeans_bankrupt_labels = kmeans_bankrupt.fit_predict(bankrupt_X_scaled)

# Gaussian Mixture Model (EM)
gmm_cancer = GaussianMixture(n_components=n_clusters, random_state=42)
gmm_cancer_labels = gmm_cancer.fit_predict(cancer_X_scaled)

gmm_bankrupt = GaussianMixture(n_components=n_clusters, random_state=42)
gmm_bankrupt_labels = gmm_bankrupt.fit_predict(bankrupt_X_scaled)

# Silhouette Scores
print("Silhouette (KMeans, Cancer):", silhouette_score(cancer_X_scaled, kmea
print("Silhouette (GMM, Cancer):", silhouette_score(cancer_X_scaled, gmm_car
```

```
print("Silhouette (KMeans, Bankrupt):", silhouette_score(bankrupt_X_scaled, 
print("Silhouette (GMM, Bankrupt):", silhouette_score(bankrupt_X_scaled, gmm
```

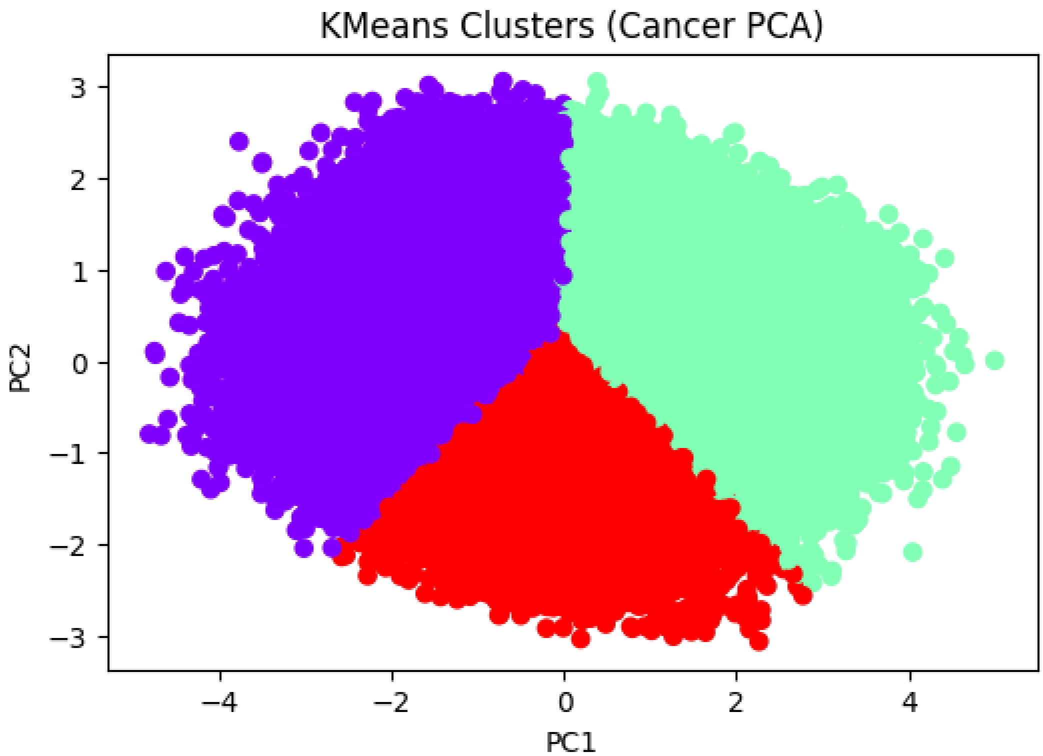
```
Silhouette (KMeans, Cancer): 0.08681559132410345
Silhouette (GMM, Cancer): 0.07222682244912479
Silhouette (KMeans, Bankrupt): 0.11391194021963115
Silhouette (GMM, Bankrupt): 0.03379523488318877
```

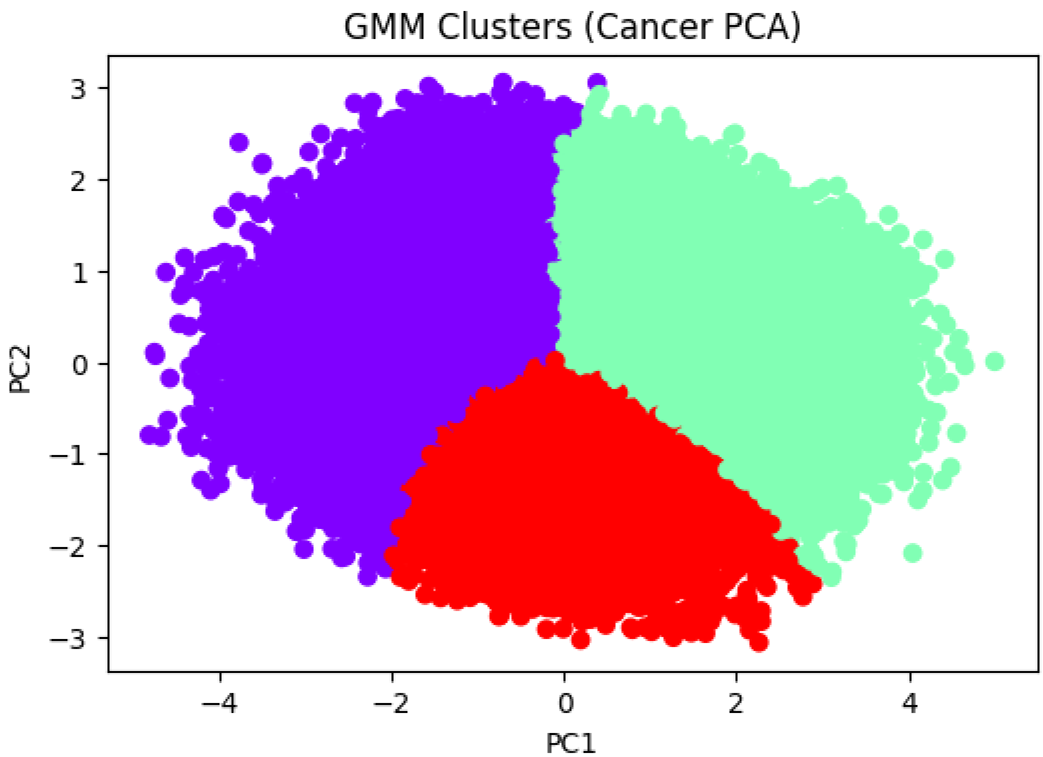
In [4]:
```
from sklearn.decomposition import PCA, FastICA
from sklearn.random_projection import GaussianRandomProjection

n_components = 2

# PCA
pca_cancer = PCA(n_components=n_components)
cancer_pca = pca_cancer.fit_transform(cancer_X_scaled)

pca_bankrupt = PCA(n_components=n_components)
bankrupt_pca = pca_bankrupt.fit_transform(bankrupt_X_scaled)

# ICA
ica_cancer = FastICA(n_components=n_components, random_state=42)
cancer_ica = ica_cancer.fit_transform(cancer_X_scaled)

ica_bankrupt = FastICA(n_components=n_components, random_state=42)
bankrupt_ica = ica_bankrupt.fit_transform(bankrupt_X_scaled)

# Randomized Projections
rp_cancer = GaussianRandomProjection(n_components=n_components, random_state
cancer_rp = rp_cancer.fit_transform(cancer_X_scaled)

rp_bankrupt = GaussianRandomProjection(n_components=n_components, random_sta
bankrupt_rp = rp_bankrupt.fit_transform(bankrupt_X_scaled)
```

In [5]:
```
# Defining function to run on
def run_clustering(X, n_clusters=3):
    kmeans = KMeans(n_clusters=n_clusters, random_state=42).fit_predict(X)
    gmm = GaussianMixture(n_components=n_clusters, random_state=42).fit_pred
    return kmeans, gmm

# Cancer dataset
km_pca_cancer, gm_pca_cancer = run_clustering(cancer_pca)
km_ica_cancer, gm_ica_cancer = run_clustering(cancer_ica)
km_rp_cancer,  gm_rp_cancer  = run_clustering(cancer_rp)

# Bankruptcy dataset
km_pca_bankrupt, gm_pca_bankrupt = run_clustering(bankrupt_pca)
km_ica_bankrupt, gm_ica_bankrupt = run_clustering(bankrupt_ica)
km_rp_bankrupt,  gm_rp_bankrupt  = run_clustering(bankrupt_rp)
```

In [6]:
```
import matplotlib.pyplot as plt

# Visualize clustering for Cancer dataset (PCA)
plt.figure(figsize=(6,4))
plt.scatter(cancer_pca[:, 0], cancer_pca[:, 1], c=km_pca_cancer, cmap='rainb
plt.title("KMeans Clusters (Cancer PCA)")
```

```
plt.xlabel("PC1")
plt.ylabel("PC2")
plt.show()

# Visualize GMM for Cancer dataset (PCA)
plt.figure(figsize=(6,4))
plt.scatter(cancer_pca[:, 0], cancer_pca[:, 1], c=gm_pca_cancer, cmap='rainb
plt.title("GMM Clusters (Cancer PCA)")
plt.xlabel("PC1")
plt.ylabel("PC2")
plt.show()

# Repeat for ICA and RP, and for bankruptcy dataset as desired
```





In [7]:
```python
# Example: Print silhouette scores
methods = {
    'Original': (cancer_X_scaled, kmeans_cancer_labels, gmm_cancer_labels),
    'PCA':      (cancer_pca, km_pca_cancer, gm_pca_cancer),
    'ICA':      (cancer_ica, km_ica_cancer, gm_ica_cancer),
    'RP':       (cancer_rp, km_rp_cancer, gm_rp_cancer)
}

for name, (X, kmeans_labels, gmm_labels) in methods.items():
    print(f"Cancer {name} - KMeans Silhouette: {silhouette_score(X, kmeans_l
    print(f"Cancer {name} - GMM Silhouette: {silhouette_score(X, gmm_labels)
```

```
Cancer Original - KMeans Silhouette: 0.087
Cancer Original - GMM Silhouette: 0.072
Cancer PCA - KMeans Silhouette: 0.329
Cancer PCA - GMM Silhouette: 0.330
Cancer ICA - KMeans Silhouette: 0.337
Cancer ICA - GMM Silhouette: 0.336
Cancer RP - KMeans Silhouette: 0.360
Cancer RP - GMM Silhouette: 0.360
```

In [8]:
```python
import pandas as pd
import numpy as np

# Load and clean
cancer_df = pd.read_csv('cancer.csv').dropna()
# Keep only numeric columns
cancer_X = cancer_df.select_dtypes(include=[np.number])
```

In [9]:
```python
n_cancer, d_cancer = cancer_X.shape
print(f"n_cancer: {n_cancer}, d_cancer: {d_cancer}")
```

```
n_cancer: 50000, d_cancer: 10
```

In [10]:
```
cancer_X.shape
```

Out[10]:
```
(50000, 10)
```

In [11]:
```
cancer_X
```

Out[11]:

	Age	Year	Genetic_Risk	Air_Pollution	Alcohol_Use	Smoking	Obesity_Level
0	71	2021	6.4	2.8	9.5	0.9	8.7
1	34	2021	1.3	4.5	3.7	3.9	6.3
2	80	2023	7.4	7.9	2.4	4.7	0.1
3	40	2015	1.7	2.9	4.8	3.5	2.7
4	43	2017	5.1	2.8	2.3	6.7	0.5
...	...	...	...	...	...	...	...
49995	80	2023	2.3	7.5	2.8	3.8	2.9
49996	40	2018	6.4	3.5	2.9	9.0	9.8
49997	74	2015	6.2	1.6	8.7	4.7	4.0
49998	21	2018	4.0	6.5	7.6	8.6	8.1
49999	22	2023	5.1	9.8	3.2	0.0	0.7

50000 rows × 10 columns

In [12]:
```python
import numpy as np
import pandas as pd

bankrupt_df = pd.read_csv('bankrupt.csv').dropna()
bankrupt_X = bankrupt_df.select_dtypes(include=[np.number])

# sample and feature counts:
n_bankrupt, d_bankrupt = bankrupt_X.shape
print(f"n_bankrupt: {n_bankrupt}, d_bankrupt: {d_bankrupt}")
```

```
n_bankrupt: 6819, d_bankrupt: 96
```

In [13]:
```python
n_bankrupt, d_bankrupt = bankrupt_X.shape
print(f"n_bankrupt: {n_bankrupt}, d_bankrupt: {d_bankrupt}")
```

```
n_bankrupt: 6819, d_bankrupt: 96
```

In [14]:
```python
import pandas as pd
from sklearn.preprocessing import StandardScaler

# Load data
cancer_df = pd.read_csv('cancer.csv')

# Drop rows with missing values
cancer_df_clean = cancer_df.dropna()
```

```python
# Select only numerical columns
cancer_X = cancer_df_clean.select_dtypes(include=['float64', 'int64'])

# Standardize features (important for PCA/KMeans)
scaler = StandardScaler()
cancer_X_scaled = scaler.fit_transform(cancer_X)
```

In [15]:
```python
from sklearn.decomposition import PCA

# Fit PCA with 2 components
pca = PCA(n_components=2)
cancer_pca = pca.fit_transform(cancer_X_scaled)

# Explained variance
explained_var = pca.explained_variance_ratio_
total_explained = explained_var.sum() * 100
print(f"Total variance explained by first two PCs: {total_explained:.1f}%")
```

```
Total variance explained by first two PCs: 30.3%
```

In [ ]:

In [ ]:

In [ ]:

In [17]:
```python
# Neural Network
```

In [18]:
```python
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.decomposition import PCA, FastICA
from sklearn.random_projection import GaussianRandomProjection
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score, f1_score

# Load and clean data
cancer_df = pd.read_csv('cancer.csv')
cancer_df_clean = cancer_df.dropna()
cancer_X = cancer_df_clean.select_dtypes(include=[np.number])

# Use Cancer_Type as label
y = cancer_df_clean['Cancer_Type'].values
le = LabelEncoder()
y_encoded = le.fit_transform(y)

# Standardize features
scaler = StandardScaler()
cancer_X_scaled = scaler.fit_transform(cancer_X)

# Dimensionality reduction
pca = PCA(n_components=2)
cancer_pca = pca.fit_transform(cancer_X_scaled)
```

```
ica = FastICA(n_components=2, random_state=42)
cancer_ica = ica.fit_transform(cancer_X_scaled)

rp = GaussianRandomProjection(n_components=2, random_state=42)
cancer_rp = rp.fit_transform(cancer_X_scaled)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(cancer_X_scaled, y_encod
X_train_pca, X_test_pca, _, _ = train_test_split(cancer_pca, y_encoded, test
X_train_ica, X_test_ica, _, _ = train_test_split(cancer_ica, y_encoded, test
X_train_rp, X_test_rp, _, _ = train_test_split(cancer_rp, y_encoded, test_si

# Neural network
def run_nn(X_train, X_test, y_train, y_test, name=""):
    clf = MLPClassifier(hidden_layer_sizes=(32,), max_iter=300, random_state
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred, average='weighted')
    print(f"{name} - Accuracy: {acc:.3f}, F1: {f1:.3f}")

run_nn(X_train, X_test, y_train, y_test, "Original")
run_nn(X_train_pca, X_test_pca, y_train, y_test, "PCA")
run_nn(X_train_ica, X_test_ica, y_train, y_test, "ICA")
run_nn(X_train_rp, X_test_rp, y_train, y_test, "RP")
```

```
Original - Accuracy: 0.123, F1: 0.120
PCA - Accuracy: 0.124, F1: 0.112
ICA - Accuracy: 0.128, F1: 0.107
RP - Accuracy: 0.124, F1: 0.109
```