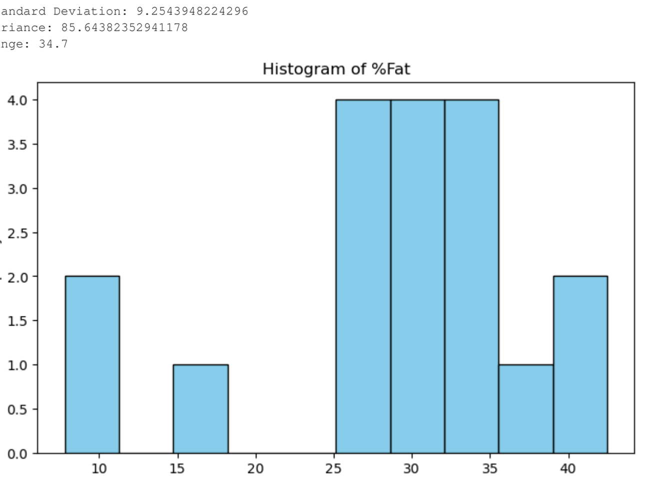
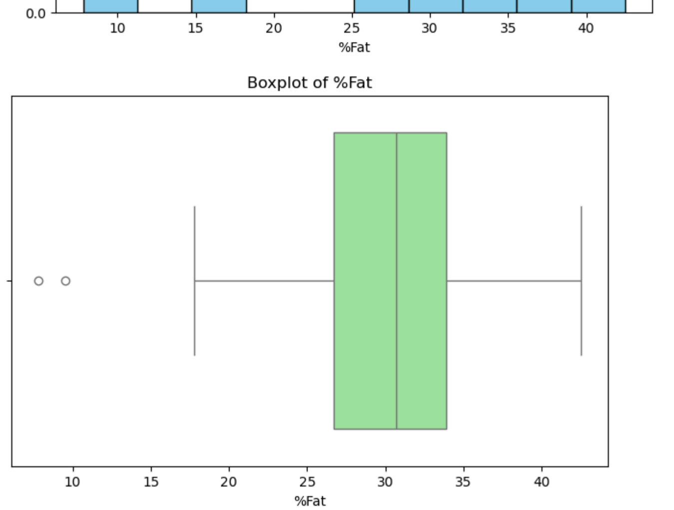
EXP NO. 1

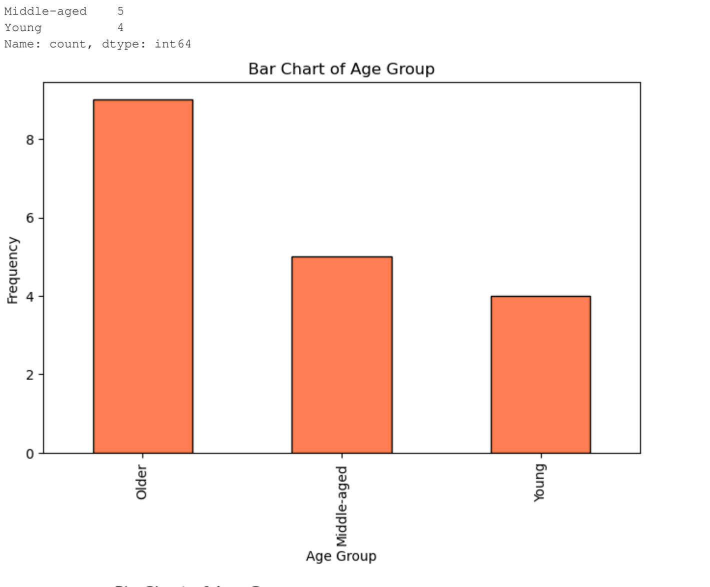
In [3]:
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
data = pd.read_csv('Age_Fat.csv')
print("Dataset Preview:\n", data.head())
fat = data['%Fat']
print(f"\nStatistics for Numerical Column:\nMean: {fat.mean()}\nMedian: {fat.median()}\nMode: {fat.mode()[0]}\nStandard Deviation: {fat.std()}\nVariance: {fat.var()}\nRange: {fat.max() - fat.min()}")
plt.figure(figsize=(8,5))
plt.hist(fat, bins=10, color='skyblue', edgecolor='black')
plt.title("Histogram of %Fat"); plt.xlabel("%Fat"); plt.ylabel("Frequency"); plt.show()
plt.figure(figsize=(8,5))
sns.boxplot(x=fat, color='lightgreen')
plt.title("Boxplot of %Fat"); plt.show()
q1, q3 = fat.quantile([0.25, 0.75])
iqr = q3 - q1
outliers = fat[(fat < q1 - 1.5*iqr) | (fat > q3 + 1.5*iqr)]
print("\nOutliers:\n", outliers)
data['Age Group'] = data['Age'].apply(lambda x: 'Young' if x < 30 else 'Middle-aged' if x <= 50 else 'Older')
counts = data['Age Group'].value_counts()
print("\nCategory Frequencies:\n", counts)
counts.plot(kind='bar', color='coral', edgecolor='black', figsize=(8,5))
plt.title("Bar Chart of Age Group"); plt.xlabel("Age Group"); plt.ylabel("Frequency"); plt.show()
counts.plot(kind='pie', autopct='%1.1f%%', startangle=90, colors=sns.color_palette('pastel'), figsize=(8,5))
plt.title("Pie Chart of Age Group"); plt.ylabel(""); plt.show()
```

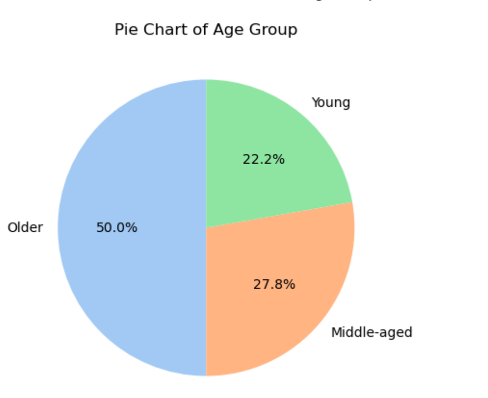
```
Dataset Preview:
    Age  %Fat
0   23   7.8
1   23   9.5
2   27  17.8
3   27  25.9
4   39  26.5

Statistics for Numerical Column:
Mean: 28.783333333333328
Median: 30.7
Mode: 7.8
Standard Deviation: 9.2543948224296
Variance: 85.64382352941178
Range: 34.7
```





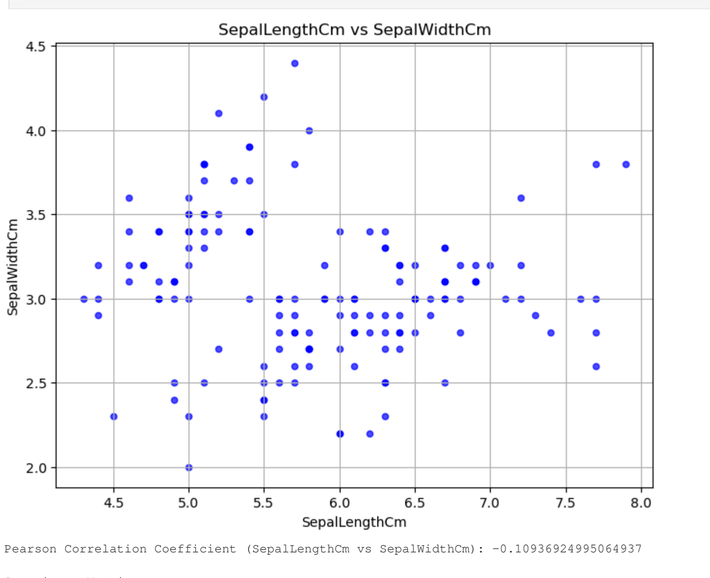

```
Outliers:
 0    7.8
1    9.5
Name: %Fat, dtype: float64

Category Frequencies:
 Age Group
Older          9
Middle-aged    5
Young          4
Name: count, dtype: int64
```



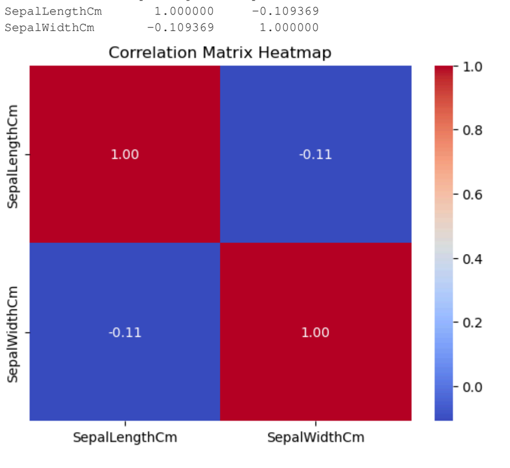


EXP NO. 2

In [11]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
df = pd.read_csv("iris.csv")
x, y = "SepalLengthCm", "SepalWidthCm"
df.plot.scatter(x=x, y=y, color="blue", alpha=0.7, figsize=(8, 6), title=f"{x} vs {y}")
plt.grid(True)
plt.show()
print(f"Pearson Correlation Coefficient ({x} vs {y}): {np.corrcoef(df[x], df[y])[0, 1]}")
print("\nCovariance Matrix:\n", np.cov(df[x], df[y]))
print("\nCorrelation Matrix:\n", df[[x, y]].corr())
sns.heatmap(df[[x, y]].corr(), annot=True, cmap="coolwarm", fmt=".2f", cbar=True)
plt.title("Correlation Matrix Heatmap")
plt.show()
```



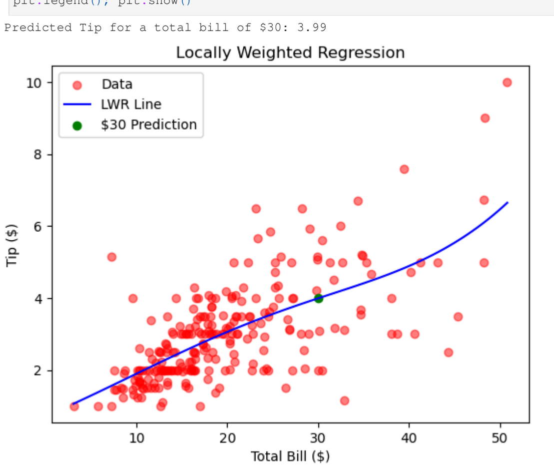

```
Pearson Correlation Coefficient (SepalLengthCm vs SepalWidthCm): -0.10936924995064937

Covariance Matrix:
 [[ 0.68569351 -0.03926846]
 [-0.03926846  0.18800403]]

Correlation Matrix:
               SepalLengthCm  SepalWidthCm
SepalLengthCm       1.000000     -0.109369
SepalWidthCm       -0.109369      1.000000
```



EXP NO. 3

In [15]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
tips = pd.read_csv("10-dataset.csv")
X, y = tips["total_bill"].values, tips["tip"].values
def lwr(xq, X, y, tau):
    w = np.exp(-((X - xq) ** 2) / (2 * tau ** 2))
    X_b = np.c_[np.ones(len(X)), X]; W = np.diag(w)
    return np.array([1, xq]) @ np.linalg.pinv(X_b.T @ W @ X_b) @ (X_b.T @ W @ y)
tau, xq = 10, 30
pred = lwr(xq, X, y, tau)
print(f"Predicted Tip for a total bill of $30: {pred:.2f}")
x_vals = np.linspace(X.min(), X.max(), 100)
y_vals = [lwr(x, X, y, tau) for x in x_vals]
plt.scatter(X, y, c='r', alpha=0.5, label="Data"); plt.plot(x_vals, y_vals, c='b', label="LWR Line")
plt.scatter([xq], [pred], c='g', label="$30 Prediction")
plt.xlabel("Total Bill ($)"); plt.ylabel("Tip ($)"); plt.title("Locally Weighted Regression")
plt.legend(); plt.show()
```

```
Predicted Tip for a total bill of $30: 3.99
```



EXP NO. 4

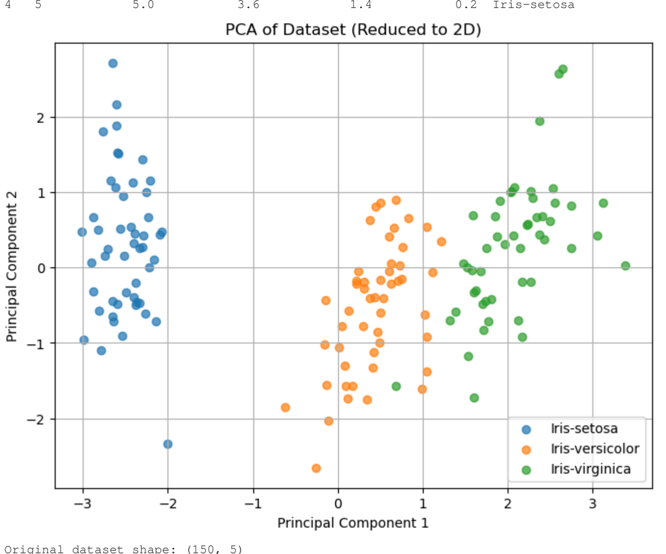
In [18]:
```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, f1_score, confusion_matrix
df = pd.read_csv('IRIS.csv')
X = StandardScaler().fit_transform(df.iloc[:, :-1])
y = LabelEncoder().fit_transform(df.iloc[:, -1])
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
def run_knn(k_values, weighted):
    for k in k_values:
        model = KNeighborsClassifier(n_neighbors=k, weights='distance' if weighted else 'uniform').fit(X_train, y_train)
        y_pred = model.predict(X_test)
        print(f"\nk={k}: Accuracy={accuracy_score(y_test, y_pred):.4f}, F1 Score={f1_score(y_test, y_pred, average='weighted'):.4f}")
        print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Regular k-NN Results:"); run_knn([1, 3, 5], weighted=False)
print("\nWeighted k-NN Results:"); run_knn([1, 3, 5], weighted=True)
```

```
Regular k-NN Results:

k=1: Accuracy=1.0000, F1 Score=1.0000
Confusion Matrix:
 [[19  0  0]
 [ 0 13  0]
 [ 0  0 13]]

k=3: Accuracy=1.0000, F1 Score=1.0000
Confusion Matrix:
 [[19  0  0]
 [ 0 13  0]
 [ 0  0 13]]

k=5: Accuracy=1.0000, F1 Score=1.0000
Confusion Matrix:
 [[19  0  0]
 [ 0 13  0]
 [ 0  0 13]]

Weighted k-NN Results:

k=1: Accuracy=1.0000, F1 Score=1.0000
Confusion Matrix:
 [[19  0  0]
 [ 0 13  0]
 [ 0  0 13]]

k=3: Accuracy=1.0000, F1 Score=1.0000
Confusion Matrix:
 [[19  0  0]
 [ 0 13  0]
 [ 0  0 13]]

k=5: Accuracy=1.0000, F1 Score=1.0000
Confusion Matrix:
 [[19  0  0]
 [ 0 13  0]
 [ 0  0 13]]
```

EXP NO. 5

In [21]:
```python
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import numpy as np
df = pd.read_csv('iris.csv')
print("Dataset preview:\n", df.head())
X = StandardScaler().fit_transform(df.iloc[:, :-1])
y = df.iloc[:, -1].values
pca = PCA(2)
X_pca = pca.fit_transform(X)
plt.figure(figsize=(8,6))
for label in np.unique(y):
    plt.scatter(X_pca[y==label, 0], X_pca[y==label, 1], label=label, alpha=0.7)
plt.title('PCA of Dataset (Reduced to 2D)')
plt.xlabel('Principal Component 1'); plt.ylabel('Principal Component 2')
plt.legend(); plt.grid(True); plt.show()
print("\nOriginal dataset shape:", X.shape)
print("Reduced dataset shape:", X_pca.shape)
print("Explained variance ratio:", pca.explained_variance_ratio_)
```

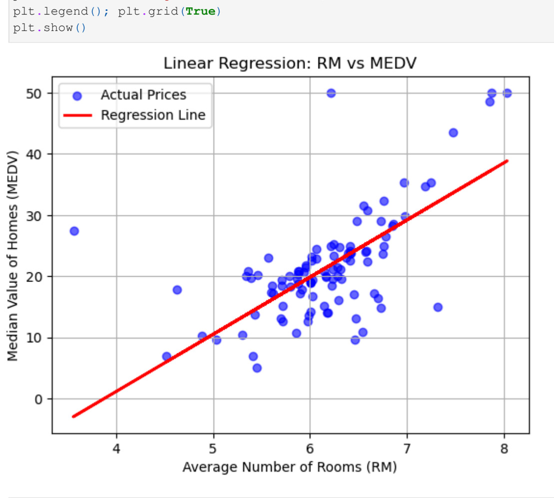
```
Dataset preview:
    Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm      Species
0   1            5.1           3.5            1.4           0.2  Iris-setosa
1   2            4.9           3.0            1.4           0.2  Iris-setosa
2   3            4.7           3.2            1.3           0.2  Iris-setosa
3   4            4.6           3.1            1.5           0.2  Iris-setosa
4   5            5.0           3.6            1.4           0.2  Iris-setosa
```



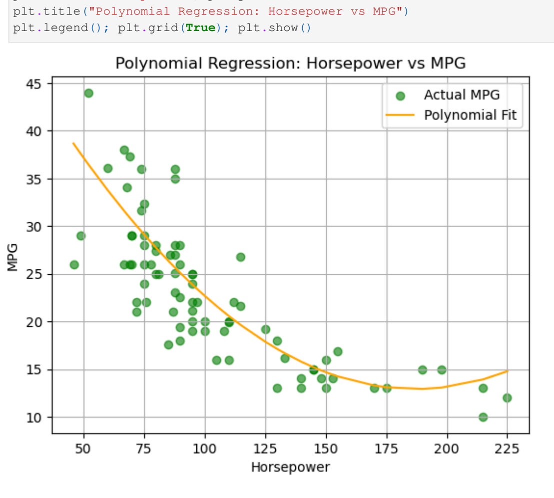

```
Original dataset shape: (150, 5)
Reduced dataset shape: (150, 2)
Explained variance ratio: [0.7470533  0.18435257]
```

EXP NO. 6

In [24]:
```python
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
data = pd.read_csv("BostonHousing.csv")
X = data[['rm']]
y = data['medv']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = LinearRegression().fit(X_train, y_train)
y_pred = model.predict(X_test)
plt.scatter(X_test, y_test, color='blue', alpha=0.6, label='Actual Prices')
plt.plot(X_test, y_pred, color='red', linewidth=2, label='Regression Line')
plt.xlabel("Average Number of Rooms (RM)"); plt.ylabel("Median Value of Homes (MEDV)")
plt.title("Linear Regression: RM vs MEDV")
plt.legend(); plt.grid(True)
plt.show()
```

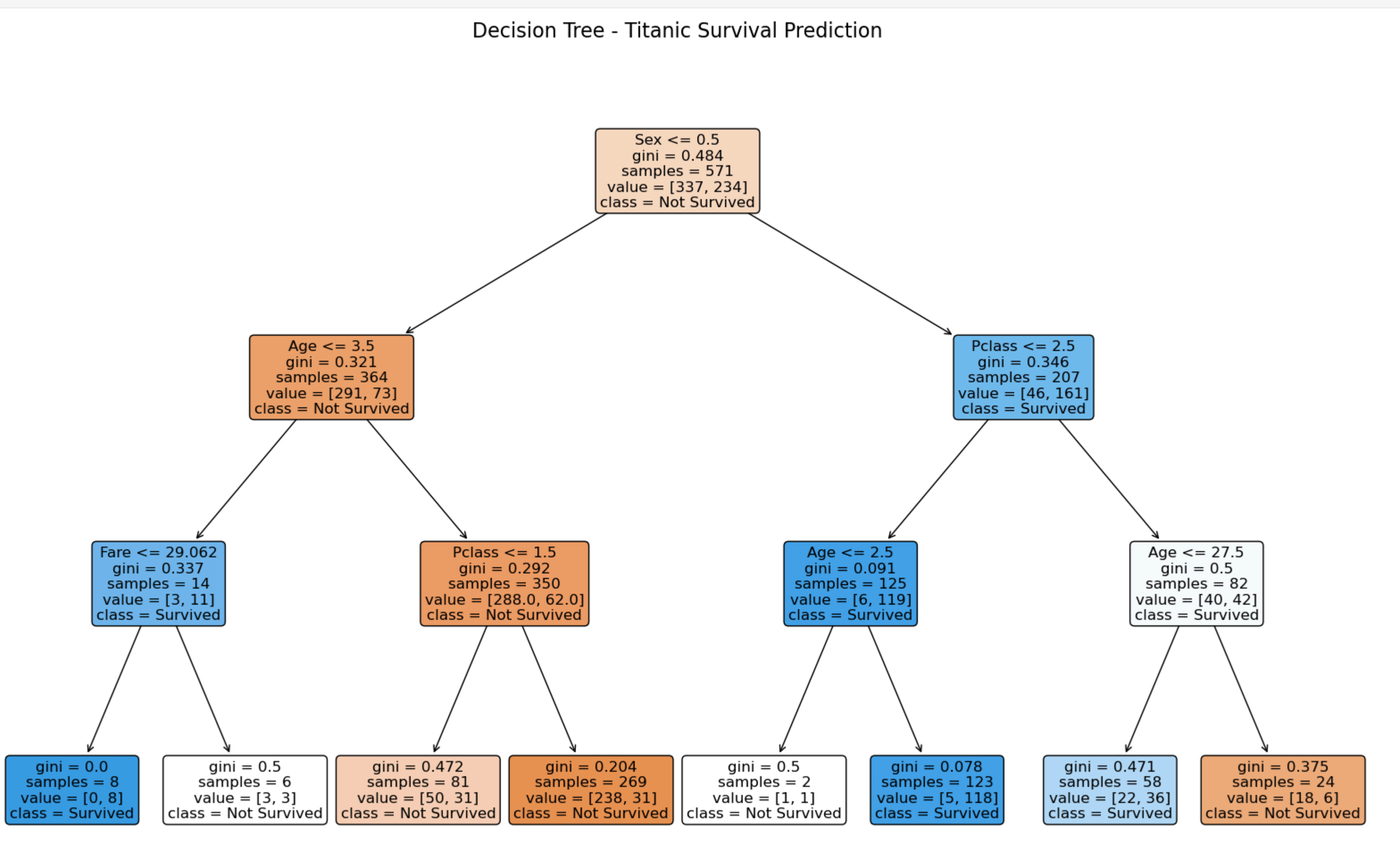


In [26]:
```python
import pandas as pd, numpy as np, matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.model_selection import train_test_split
data = pd.read_csv("mpg.csv")
data.replace('?', np.nan, inplace=True)
data.dropna(subset=['horsepower'], inplace=True)
data['horsepower'] = data['horsepower'].astype(float)
X, y = data[['horsepower']], data['mpg']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
poly = PolynomialFeatures(2)
X_train_poly, X_test_poly = poly.fit_transform(X_train), poly.transform(X_test)
model = LinearRegression().fit(X_train_poly, y_train)
y_pred = model.predict(X_test_poly)
plt.scatter(X_test, y_test, c='green', alpha=0.6, label='Actual MPG')
idx = X_test['horsepower'].argsort()
plt.plot(X_test.iloc[idx], y_pred[idx], c='orange', label='Polynomial Fit')
plt.xlabel("Horsepower"); plt.ylabel("MPG")
plt.title("Polynomial Regression: Horsepower vs MPG")
plt.legend(); plt.grid(True); plt.show()
```



EXP NO. 7

In [29]:
```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
import matplotlib.pyplot as plt
df = pd.read_csv('titanic.csv')[['Survived', 'Pclass', 'Sex', 'Age', 'Fare']].dropna()
df['Sex'] = df['Sex'].map({'male': 0, 'female': 1})
X, y = df.drop('Survived', axis=1), df['Survived']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
clf = DecisionTreeClassifier(max_depth=3, random_state=42).fit(X_train, y_train)
plt.figure(figsize=(20, 12))
plot_tree(clf, feature_names=X.columns, class_names=["Not Survived", "Survived"],filled=True, rounded=True, fontsize=12)
plt.title("Decision Tree - Titanic Survival Prediction\n", fontsize=16)
plt.show()
y_pred = clf.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Precision:", precision_score(y_test, y_pred))
print("Recall:", recall_score(y_test, y_pred))
print("F1-score:", f1_score(y_test, y_pred))
```




```
Accuracy: 0.7412587412587412
Precision: 0.6938775510204082
Recall: 0.6071428571428571
F1-score: 0.6476190476190476
```

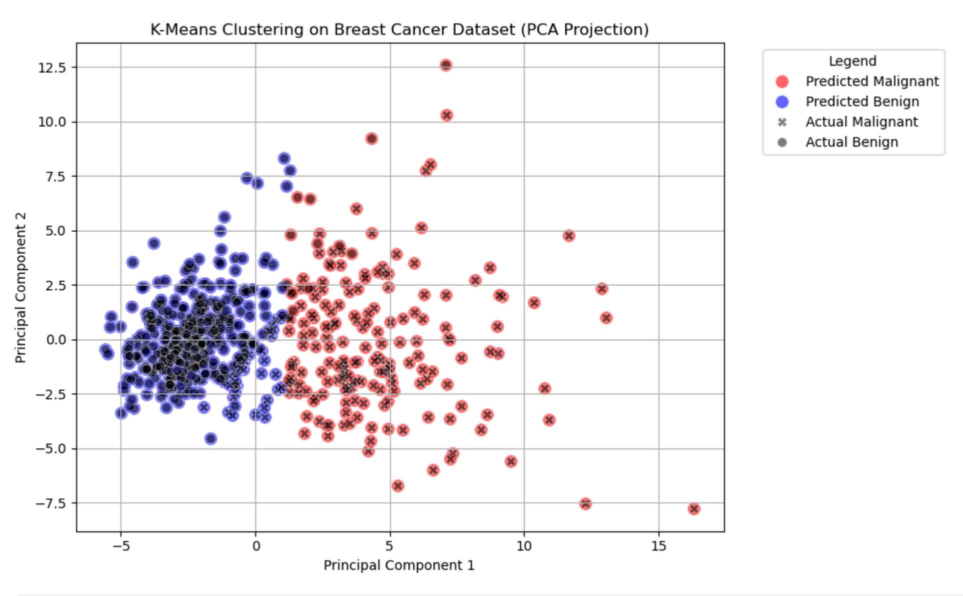
EXP NO. 8

In [32]:
```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
data = pd.read_csv('iris.csv')
X_train, X_test, y_train, y_test = train_test_split(data.iloc[:, :-1], data.iloc[:, -1], test_size=0.3, random_state=42)
y_pred = GaussianNB().fit(X_train, y_train).predict(X_test)
print(f"Accuracy of the Naive Bayes Classifier: {accuracy_score(y_test, y_pred) * 100:.2f}%")
```

```
Accuracy of the Naive Bayes Classifier: 100.00%
```

EXP NO. 9

In [35]:
```python
import pandas as pd, matplotlib.pyplot as plt, seaborn as sns
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from scipy.stats import mode
df = pd.read_csv("Breast Cancer Wisconsin.csv").drop(columns=['id','Unnamed: 32'], errors='ignore')
diagnosis = df.pop('diagnosis').map({'M':1,'B':0}) if 'diagnosis' in df else None
scaled = StandardScaler().fit_transform(df)
labels = KMeans(2, random_state=42).fit_predict(scaled)
pca = PCA(2).fit_transform(scaled)
vis_df = pd.DataFrame({'PCA1':pca[:,0],'PCA2':pca[:,1],'Cluster':labels})
if diagnosis is not None:
    vis_df['Actual'] = diagnosis
    cluster_map = {c:'Predicted Malignant' if mode(diagnosis[labels==c]).mode==1 else 'Predicted Benign' for c in [0,1]}
    vis_df['Cluster_Label'] = vis_df['Cluster'].map(cluster_map)
    vis_df['Actual_Label'] = vis_df['Actual'].map({1:'Actual Malignant',0:'Actual Benign'})
plt.figure(figsize=(10,6))
sns.scatterplot(data=vis_df, x='PCA1', y='PCA2', hue='Cluster_Label',palette={'Predicted Malignant':'red','Predicted Benign':'blue'}, s=100, alpha=0.6)
sns.scatterplot(data=vis_df, x='PCA1', y='PCA2', style='Actual_Label',markers={'Actual Malignant':'X','Actual Benign':'o'}, color='black', s=50, alpha=0.5)
plt.title("K-Means Clustering on Breast Cancer Dataset (PCA Projection)")
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.legend(title="Legend", bbox_to_anchor=(1.05,1), loc='upper left')
plt.grid(True)
plt.tight_layout()
plt.show()
```


K-Means Clustering on Breast Cancer Dataset (PCA Projection)
Legend
Predicted Malignant
Predicted Benign
Actual Malignant
Actual Benign
Principal Component 1
Principal Component 2
12.5
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−5
0
5
10
15

In [ ]: