In [1]:

```
#BL.EN.U4CSE21083
#K.SAI CHANDANA
#CSEB
import pandas as pd
import numpy as np
df=pd.read_csv('winequality-red.csv')
df
```

Out[1]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56
...	...	...	...	...	...	...	...	...	...	...
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66

1599 rows × 12 columns

In [2]:

```
#A1
# Sample data for two classes (quality 5 and quality 6)
class_5_data = np.array([
    [7.4, 0.7, 0, 1.9, 0.076, 11, 34, 0.9978, 3.51, 0.56, 9.4],
    [7.4, 0.66, 0, 1.8, 0.075, 13, 40, 0.9978, 3.51, 0.56, 9.4]
])

class_6_data = np.array([
    [11.2, 0.28, 0.56, 1.9, 0.075, 17, 60, 0.998, 3.16, 0.58, 9.8],
    [7.9, 0.32, 0.51, 1.8, 0.341, 17, 56, 0.9969,3.04, 1.08, 9.2]
])

# Calculating mean vectors
mean_class_5 = np.mean(class_5_data, axis=0)
mean_class_6 = np.mean(class_6_data, axis=0)

# Calculating standard deviations(spread)
std_class_5 = np.std(class_5_data, axis=0)
std_class_6 = np.std(class_6_data, axis=0)

# Calculate Euclidean distance between mean vectors
distance = np.linalg.norm(mean_class_5 - mean_class_6)

print("Mean vector for class 5:", mean_class_5)
print("Mean vector for class 6:", mean_class_6)
print("Standard deviation for class 5:", std_class_5)
print("Standard deviation for class 6:", std_class_6)
print("Euclidean distance between mean vectors:", distance)
```

```
Mean vector for class 5: [ 7.4     0.68    0.      1.85    0.0755 12.
37.      0.9978  3.51
  0.56    9.4   ]
Mean vector for class 6: [ 9.55     0.3      0.535    1.85     0.208   17.
58.       0.99745
  3.1      0.83     9.5     ]
Standard deviation for class 5: [0.e+00 2.e-02 0.e+00 5.e-02 5.e-04 1.e+00
3.e+00 0.e+00 0.e+00 0.e+00
 0.e+00]
Standard deviation for class 6: [1.65e+00 2.00e-02 2.50e-02 5.00e-02 1.33e
-01 0.00e+00 2.00e+00 5.50e-04
 6.00e-02 2.50e-01 3.00e-01]
Euclidean distance between mean vectors: 21.70994429685392
```

In [3]:

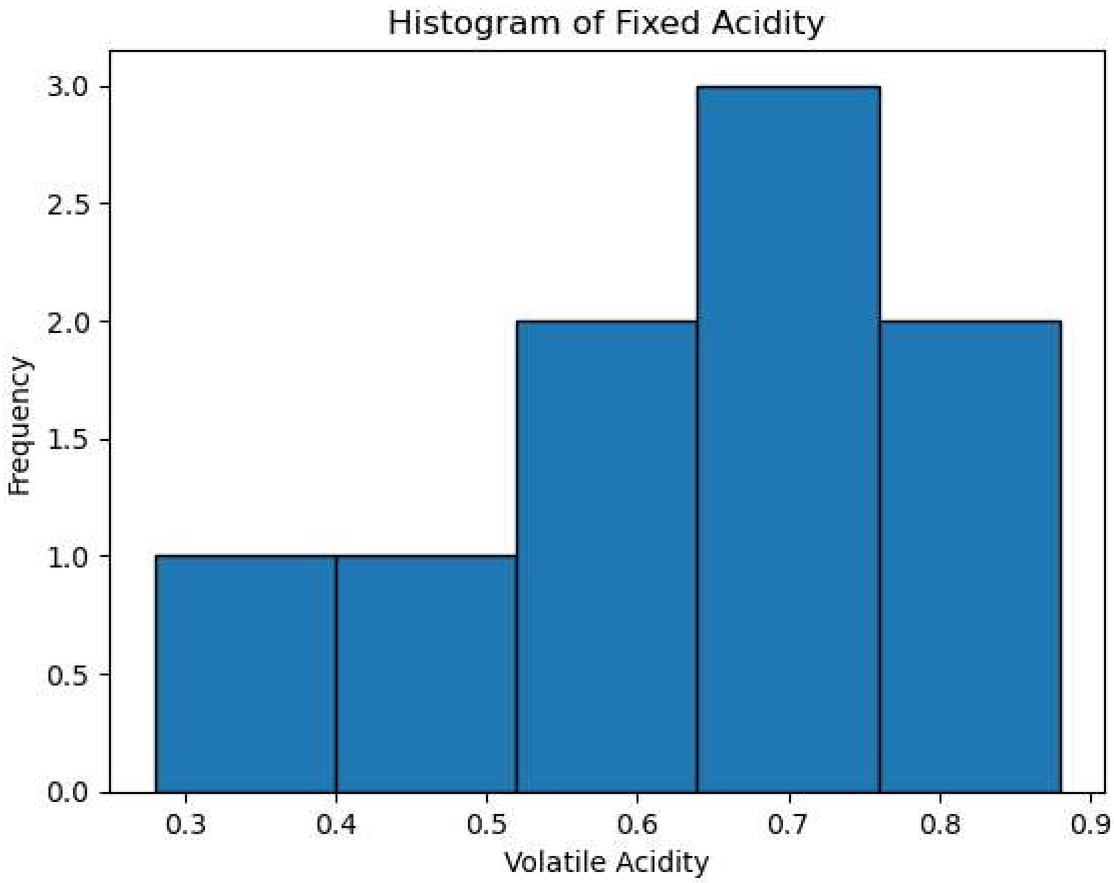
```
#A2
import matplotlib.pyplot as plt

# Sample data for volatile acidity
volatile_acidity_data = np.array([
    0.7, 0.88, 0.76, 0.28, 0.7, 0.66, 0.6, 0.58, 0.5
])

# Plotting histogram of feature volatile acidity
plt.hist(volatile_acidity_data, bins=5, edgecolor='k')
plt.xlabel("Volatile Acidity")
plt.ylabel("Frequency")
plt.title("Histogram of Fixed Acidity")
plt.show()

# Calculating mean and variance of feature volatile acidity
mean_volatile_acidity = np.mean(volatile_acidity_data)
variance_volatile_acidity = np.var(volatile_acidity_data)

print("Mean volatile acidity:", mean_volatile_acidity)
print("Variance of volatile acidity:", variance_volatile_acidity)
```




```
Mean volatile acidity: 0.6288888888888889
Variance of volatile acidity: 0.02587654320987654
```

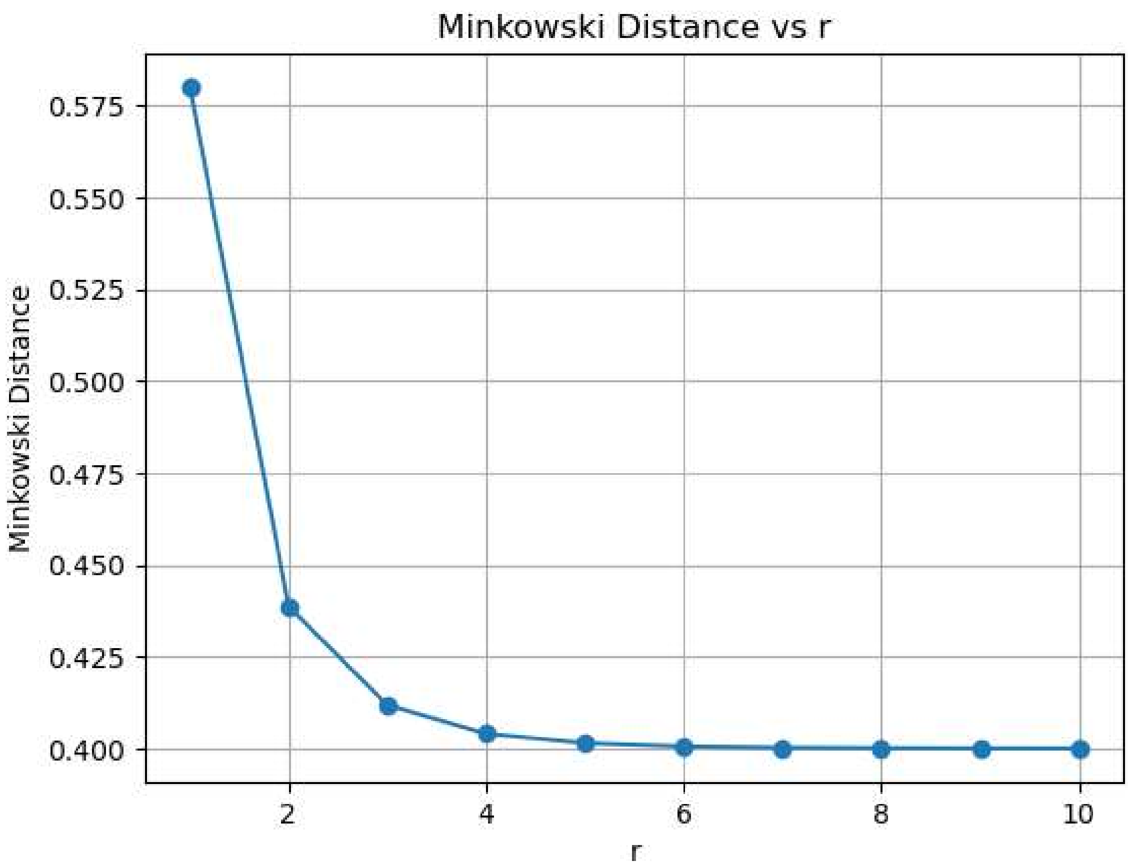
In [4]:

```
#A3
from scipy.spatial import distance

# Sample feature vectors of fixed acidity and volatile acidity
vector1 = np.array([7.4, 0.7])
vector2 = np.array([7.8, 0.88])

# Calculating Minkowski distances for different values of r
r_values = range(1, 11)
distances = [distance.minkowski(vector1, vector2, r) for r in r_values]

# Plotting the distances
plt.plot(r_values, distances, marker='o')
plt.xlabel("r")
plt.ylabel("Minkowski Distance")
plt.title("Minkowski Distance vs r")
plt.grid(True)
plt.show()
```



In [5]:

```
#A4
from sklearn.model_selection import train_test_split
# Extracting features (X) and target variable (y)
X = df.drop(columns=['quality'])
y = df['quality']
# Splitting the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
print("Printing the training dataset")
print(X)
print("Printing the testing dataset")
print(y)
```

```
Printing the training dataset
      fixed acidity  volatile acidity  citric acid  residual sugar  chlori
des  \
0               7.4             0.700         0.00             1.9      0.
076
1               7.8             0.880         0.00             2.6      0.
098
2               7.8             0.760         0.04             2.3      0.
092
3              11.2             0.280         0.56             1.9      0.
075
4               7.4             0.700         0.00             1.9      0.
076
...             ...               ...          ...             ...
...
1594            6.2             0.600         0.08             2.0      0.
090
1595            5.9             0.550         0.10             2.2      0.
062
1596            6.3             0.510         0.13             2.3      0.
076
1597            5.9             0.645         0.12             2.0      0.
075
1598            6.0             0.310         0.47             3.6      0.
067

      free sulfur dioxide  total sulfur dioxide  density    pH  sulphates
\
0                    11.0                  34.0  0.99780  3.51       0.56
1                    25.0                  67.0  0.99680  3.20       0.68
2                    15.0                  54.0  0.99700  3.26       0.65
3                    17.0                  60.0  0.99800  3.16       0.58
4                    11.0                  34.0  0.99780  3.51       0.56
...                   ...                   ...      ...   ...        ...
1594                 32.0                  44.0  0.99490  3.45       0.58
1595                 39.0                  51.0  0.99512  3.52       0.76
1596                 29.0                  40.0  0.99574  3.42       0.75
1597                 32.0                  44.0  0.99547  3.57       0.71
1598                 18.0                  42.0  0.99549  3.39       0.66

      alcohol
0         9.4
1         9.8
2         9.8
3         9.8
4         9.4
...       ...
1594     10.5
1595     11.2
1596     11.0
1597     10.2
1598     11.0

[1599 rows x 11 columns]
Printing the testing dataset
0       5
1       5
2       5
3       6
4       5
       ..
```

```
1594    5
1595    6
1596    6
1597    5
1598    6
Name: quality, Length: 1599, dtype: int64
```

In [6]:

```
#A5
from sklearn.neighbors import KNeighborsClassifier
neigh = KNeighborsClassifier(n_neighbors=3)
neigh.fit(X, y)
```

Out[6]:

```
▾       KNeighborsClassifier
KNeighborsClassifier(n_neighbors=3)
```

In [8]:

```
#A6
accuracy = neigh.score(X_test, y_test)
print(f"Accuracy on the test set: {accuracy:2f}")
```

```
Accuracy on the test set: 0.735417
```

In [9]:

```
#A7
y_pred = neigh.predict(X_test)
from sklearn.metrics import accuracy_score
accuracy= accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

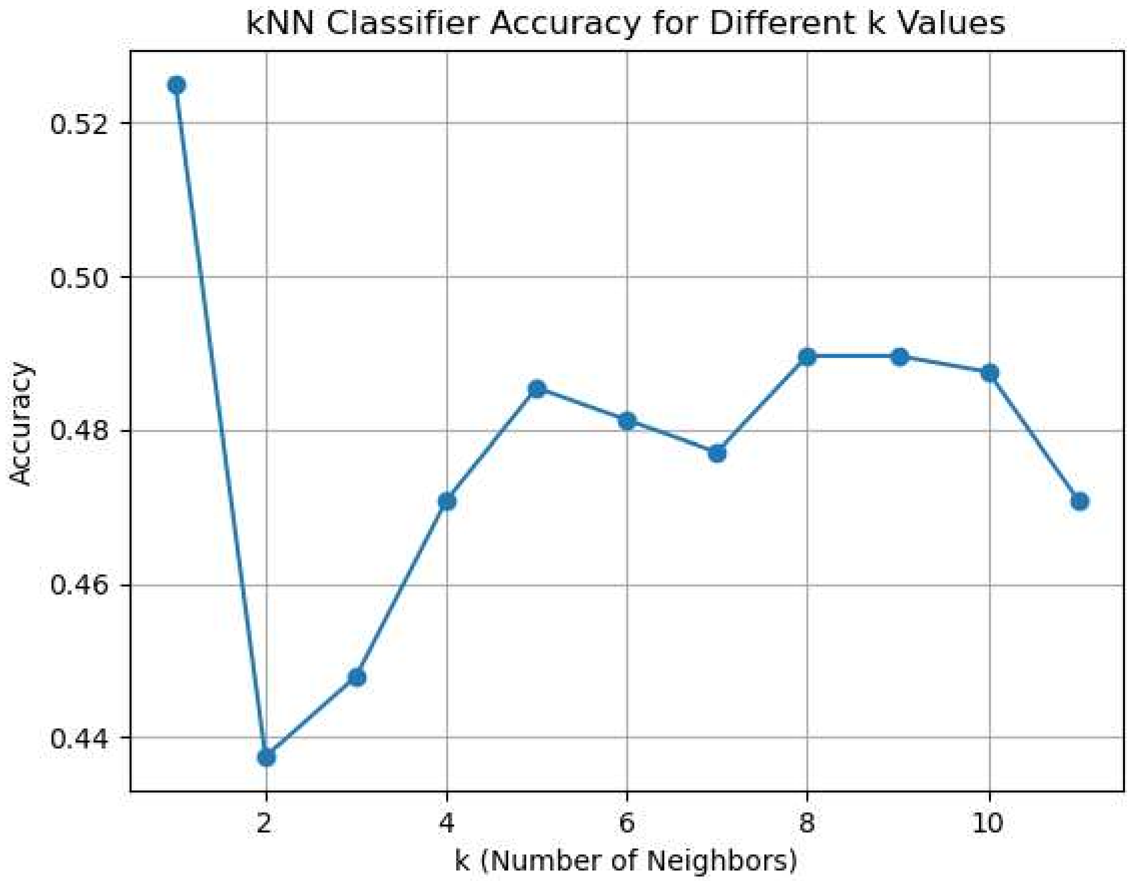
```
Accuracy: 0.7354166666666667
```

In [10]:

```
#A8
k_values = range(1, 12)
accuracies = []

for k in k_values:
    neigh = KNeighborsClassifier(n_neighbors=k)
    neigh.fit(X_train, y_train)
    accuracy = neigh.score(X_test, y_test)
    accuracies.append(accuracy)

# Plot the accuracies
plt.plot(k_values, accuracies, marker='o')
plt.xlabel("k (Number of Neighbors)")
plt.ylabel("Accuracy")
plt.title("kNN Classifier Accuracy for Different k Values")
plt.grid(True)
plt.show()
```



In [11]:

```
#A9
from sklearn.metrics import confusion_matrix, classification_report
# Calculating confusion matrix
conf_matrix = confusion_matrix(y_test, predictions)
# Generating classification report
class_report = classification_report(y_test, predictions)
print("Confusion Matrix:")
print(conf_matrix)
print("\nClassification Report:")
print(class_report)
```

```
Confusion Matrix:
[[  1   0   0   0   0   0]
 [  0   8   2   7   0   0]
 [  1   2 157  33   2   0]
 [  1   4  34 149  12   0]
 [  0   1   8  14  37   1]
 [  0   0   1   4   0   1]]

Classification Report:
              precision    recall  f1-score   support

           3       0.33      1.00      0.50         1
           4       0.53      0.47      0.50        17
           5       0.78      0.81      0.79       195
           6       0.72      0.74      0.73       200
           7       0.73      0.61      0.66        61
           8       0.50      0.17      0.25         6

    accuracy                           0.74       480
   macro avg       0.60      0.63      0.57       480
weighted avg       0.73      0.74      0.73       480
```

In [ ]: