In [1]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score,classification_report, confusion
```

In [2]:
```
sonar_data = pd.read_csv('sonar.csv', header=None)
```

In [3]:
```
sonar_data.head()
```

Out[3]:

	0	1	2	3	4	5	6	7	8	9	...	51
0	0.0200	0.0371	0.0428	0.0207	0.0954	0.0986	0.1539	0.1601	0.3109	0.2111	...	0.0027
1	0.0453	0.0523	0.0843	0.0689	0.1183	0.2583	0.2156	0.3481	0.3337	0.2872	...	0.0084
2	0.0262	0.0582	0.1099	0.1083	0.0974	0.2280	0.2431	0.3771	0.5598	0.6194	...	0.0232
3	0.0100	0.0171	0.0623	0.0205	0.0205	0.0368	0.1098	0.1276	0.0598	0.1264	...	0.0121
4	0.0762	0.0666	0.0481	0.0394	0.0590	0.0649	0.1209	0.2467	0.3564	0.4459	...	0.0031

5 rows × 61 columns

In [4]:
```
sonar_data.tail()
```

Out[4]:

	0	1	2	3	4	5	6	7	8	9	...	5
203	0.0187	0.0346	0.0168	0.0177	0.0393	0.1630	0.2028	0.1694	0.2328	0.2684	...	0.011
204	0.0323	0.0101	0.0298	0.0564	0.0760	0.0958	0.0990	0.1018	0.1030	0.2154	...	0.006
205	0.0522	0.0437	0.0180	0.0292	0.0351	0.1171	0.1257	0.1178	0.1258	0.2529	...	0.016
206	0.0303	0.0353	0.0490	0.0608	0.0167	0.1354	0.1465	0.1123	0.1945	0.2354	...	0.008
207	0.0260	0.0363	0.0136	0.0272	0.0214	0.0338	0.0655	0.1400	0.1843	0.2354	...	0.014

5 rows × 61 columns

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

Out[5]:
```
(208, 61)
```

In [6]:
```
sonar_data.groupby(60).mean()
```

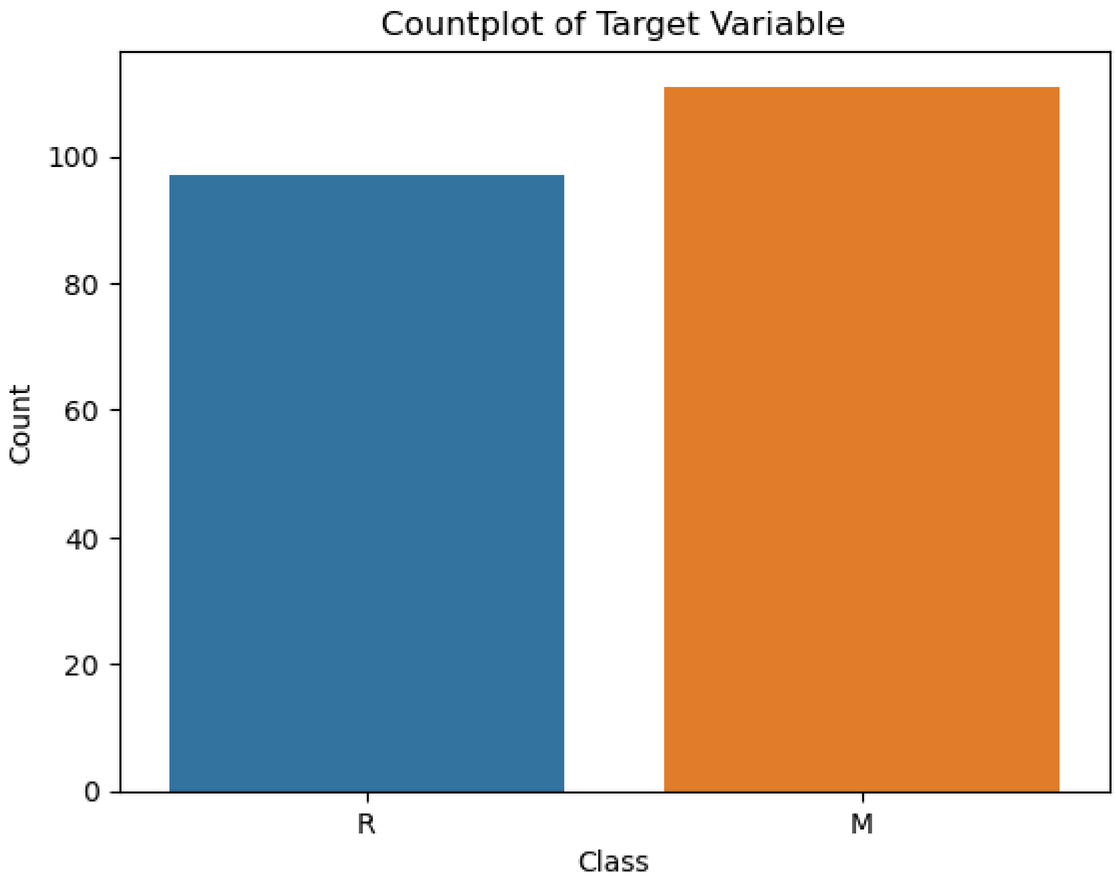
Out[6]:

60	0	1	2	3	4	5	6	7	8
M	0.034989	0.045544	0.050720	0.064768	0.086715	0.111864	0.128359	0.149832	0.213492
R	0.022498	0.030303	0.035951	0.041447	0.062028	0.096224	0.114180	0.117596	0.137392

2 rows × 60 columns

In [7]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming 'sonar_data' is your DataFrame and the target variable is in col
sns.countplot(x=sonar_data[60])
plt.title('Countplot of Target Variable')
plt.xlabel('Class')
plt.ylabel('Count')
plt.show()
```



In [8]:
```
# Separating data and labels
X = sonar_data.drop(columns=60, axis=1)
Y = sonar_data[60]
```

In [9]:
```
# Split data into training and testing sets
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.19, s
```

In [10]:
```
# Feature Scaling
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

In [11]:
```
# Logistic Regression Model Training
logistic_regression_model = LogisticRegression()
logistic_regression_model.fit(X_train, Y_train)
```

Out[11]:
```
LogisticRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [12]:
```
# Evaluate Logistic Regression Model
training_data_accuracy_lr = accuracy_score(logistic_regression_model.predic
test_data_accuracy_lr = logistic_regression_model.score(X_test, Y_test)
```

In [13]:
```
# Random Forest Model Training
random_forest_model = RandomForestClassifier(random_state=42)
random_forest_model.fit(X_train_scaled, Y_train)
```

Out[13]:
```
RandomForestClassifier(random_state=42)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [14]:
```
# Evaluate Random Forest Model
training_data_accuracy_rf = accuracy_score(random_forest_model.predict(X_tr
test_data_accuracy_rf = accuracy_score(random_forest_model.predict(X_test_s
```

In [15]:
```
print('\nLogistic Regression Model:')
print('Accuracy on training data: ', training_data_accuracy_lr)
print('Accuracy on test data: ', test_data_accuracy_lr)
```

```

Logistic Regression Model:
Accuracy on training data:  0.8095238095238095
Accuracy on test data:  0.85
```

In [16]:
```
print('\nRandom Forest Model:')
print('Accuracy on training data: ', training_data_accuracy_rf)
print('Accuracy on test data: ', test_data_accuracy_rf)
```

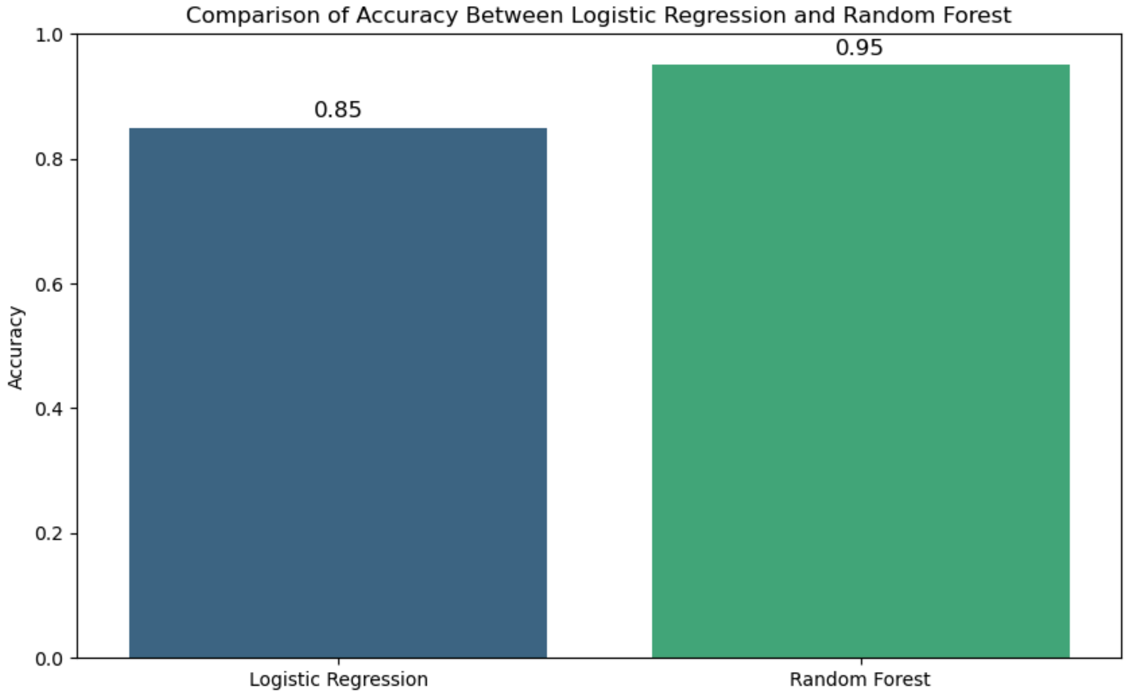
```
Random Forest Model:
Accuracy on training data:  1.0
Accuracy on test data:  0.95
```

In [17]:
```
# Comparison of Accuracy Between Logistic Regression and Random Forest
models = ['Logistic Regression', 'Random Forest']
accuracies = [test_data_accuracy_lr, test_data_accuracy_rf]

plt.figure(figsize=(10, 6))
sns.barplot(x=models, y=accuracies, palette='viridis')
plt.title('Comparison of Accuracy Between Logistic Regression and Random Fo
plt.ylabel('Accuracy')
plt.ylim(0, 1)

# Add accuracy values on top of each bar
for i, acc in enumerate(accuracies):
    plt.text(i, acc + 0.01, f'{acc:.2f}', ha='center', va='bottom', fontsiz

plt.show()
```



In [18]:

```
# Assuming 'new_data' contains 60 values for features\
new_data=(0.0119,0.0582,0.0623,0.0600,0.1397,0.1883,0.1422,0.1447,0.0487,0.

new_data_array = np.array(new_data)

# Reshape the array to be a 2D array, as the model expects a 2D input
new_data_array_reshaped = new_data_array.reshape(1, -1)

# Scale the input data using the same scaler that was used for training
scaled_new_data = scaler.transform(new_data_array_reshaped)

# Now, you can make predictions using both models
rf_prediction = random_forest_model.predict(scaled_new_data)
lr_prediction = logistic_regression_model.predict(scaled_new_data)

# Print the predictions
print('Random Forest Prediction:', rf_prediction[0])
```

```
Random Forest Prediction: R
```

In [ ]: