1. Dataset Exploration:

In [29]:

```
import pandas as pd

# Load the dataset
df = pd.read_csv('dry-beans.csv')
```

In [49]:

```
# Display the first few rows of the dataset
df.head()
```

Out[49]:

	Area	Perimeter	MajorAxisLength	MinorAxisLength	AspectRation	Eccentricity	ConvexAı
0	28395	610.291	208.178117	173.888747	1.197191	0.549812	287
1	28734	638.018	200.524796	182.734419	1.097356	0.411785	291
2	29380	624.110	212.826130	175.931143	1.209713	0.562727	296
3	30008	645.884	210.557999	182.516516	1.153638	0.498616	307
4	30140	620.134	201.847882	190.279279	1.060798	0.333680	304

In [50]:

```
# Check the dimensions of the dataset
df.shape
```

Out[50]:

```
(13611, 17)
```

In [51]:

```
# Check the data types of the columns
df.dtypes
```

Out[51]:

```
Area                int64
Perimeter         float64
MajorAxisLength   float64
MinorAxisLength   float64
AspectRation      float64
Eccentricity      float64
ConvexArea          int64
EquivDiameter     float64
Extent            float64
Solidity          float64
roundness         float64
Compactness       float64
ShapeFactor1      float64
ShapeFactor2      float64
ShapeFactor3      float64
ShapeFactor4      float64
Class              object
dtype: object
```

In [33]:

```
# Check for missing values
print(df.isnull().sum())
```

```
Area              0
Perimeter         0
MajorAxisLength   0
MinorAxisLength   0
AspectRation      0
Eccentricity      0
ConvexArea        0
EquivDiameter     0
Extent            0
Solidity          0
roundness         0
Compactness       0
ShapeFactor1      0
ShapeFactor2      0
ShapeFactor3      0
ShapeFactor4      0
Class             0
dtype: int64
```

In [52]:

```
# Descriptive statistics
df.describe()
```

Out[52]:

	Area	Perimeter	MajorAxisLength	MinorAxisLength	AspectRation	Eccer
count	13611.000000	13611.000000	13611.000000	13611.000000	13611.000000	13611.(
mean	53048.284549	855.283459	320.141867	202.270714	1.583242	0.7
std	29324.095717	214.289696	85.694186	44.970091	0.246678	0.(
min	20420.000000	524.736000	183.601165	122.512653	1.024868	0.2
25%	36328.000000	703.523500	253.303633	175.848170	1.432307	0.7
50%	44652.000000	794.941000	296.883367	192.431733	1.551124	0.7
75%	61332.000000	977.213000	376.495012	217.031741	1.707109	0.8
max	254616.000000	1985.370000	738.860154	460.198497	2.430306	0.9

In [53]:

```
# Class distribution
df['Class'].value_counts()
```

Out[53]:

```
DERMASON    3546
SIRA        2636
SEKER       2027
HOROZ       1928
CALI        1630
BARBUNYA    1322
BOMBAY       522
Name: Class, dtype: int64
```

2. Data Preprocessing:

In [36]:

```
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.model_selection import train_test_split
```

In [37]:

```
# Separate the features (X) and target variable (y)
X = df.drop('Class', axis=1)
y = df['Class']
```

In [38]:

```
# Handle missing values if any
X = X.fillna(X.mean())  # Replace missing values with column means
```

In [39]:

```
# Scale numerical features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

In [40]:

```
# Encode categorical variables
encoder = LabelEncoder()
y_encoded = encoder.fit_transform(y)
```

In [41]:

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_encoded, test_size=0.2, r
```

3. Model Training:

In [42]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
```

In [43]:

```
# Initialize the models
lr = LogisticRegression()
dt = DecisionTreeClassifier()
knn = KNeighborsClassifier()
nb = GaussianNB()
svm = SVC()
```

In [44]:

```
# Fit the models on the training data
lr.fit(X_train, y_train)
dt.fit(X_train, y_train)
knn.fit(X_train, y_train)
nb.fit(X_train, y_train)
svm.fit(X_train, y_train)
```

```
C:\Users\Asad\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.p
y:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown i
n:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sc
ikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-reg
ression (https://scikit-learn.org/stable/modules/linear_model.html#logisti
c-regression)
  n_iter_i = _check_optimize_result(
```

Out[44]:

▸ SVC

4. Model Evaluation:

In [45]:

```
from sklearn.metrics import accuracy_score

# Make predictions on the testing data
lr_pred = lr.predict(X_test)
dt_pred = dt.predict(X_test)
knn_pred = knn.predict(X_test)
nb_pred = nb.predict(X_test)
svm_pred = svm.predict(X_test)
```

In [46]:

```
# Calculate accuracy scores
lr_accuracy = accuracy_score(y_test, lr_pred)
dt_accuracy = accuracy_score(y_test, dt_pred)
knn_accuracy = accuracy_score(y_test, knn_pred)
nb_accuracy = accuracy_score(y_test, nb_pred)
svm_accuracy = accuracy_score(y_test, svm_pred)
```

In [47]:

```
# Compare the performances
print("Logistic Regression Accuracy:", lr_accuracy)
print("Decision Tree Accuracy:", dt_accuracy)
print("k-Nearest Neighbors Accuracy:", knn_accuracy)
print("Naïve Bayes Accuracy:", nb_accuracy)
print("Support Vector Machine Accuracy:", svm_accuracy)
```

```
Logistic Regression Accuracy: 0.9261843554902681
Decision Tree Accuracy: 0.8909291222915902
k-Nearest Neighbors Accuracy: 0.9232464193903782
Naïve Bayes Accuracy: 0.9037825927286082
Support Vector Machine Accuracy: 0.9338964377524789
```

5. Performance Comparison:

In [48]:

```
import matplotlib.pyplot as plt

# Create a bar chart of model accuracies
models = ['Logistic Regression', 'Decision Tree', 'k-Nearest Neighbors', 'Naïve Bayes', '
accuracies = [lr_accuracy, dt_accuracy, knn_accuracy, nb_accuracy, svm_accuracy]

plt.bar(models, accuracies)
plt.xlabel('Models')
plt.ylabel('Accuracy')
plt.title('Model Performance Comparison')
plt.xticks(rotation=45)
plt.ylim(0, 1.0)
plt.show()
```