Logistic Regression and Support Vector Classification on Abalone Dataset

This notebook will guide you through reading the data, performing exploratory data analysis (EDA), standardizing the input, and building Logistic Regression and Support Vector Classification models. We will also apply Principal Component Analysis (PCA) to determine the significant variables and rebuild the models with these components.

Step 1: Read the Data

In [1]:
```
import pandas as pd

# Load the dataset
data_path = r'dataset\abalone.csv'
abalone_df = pd.read_csv(data_path)

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

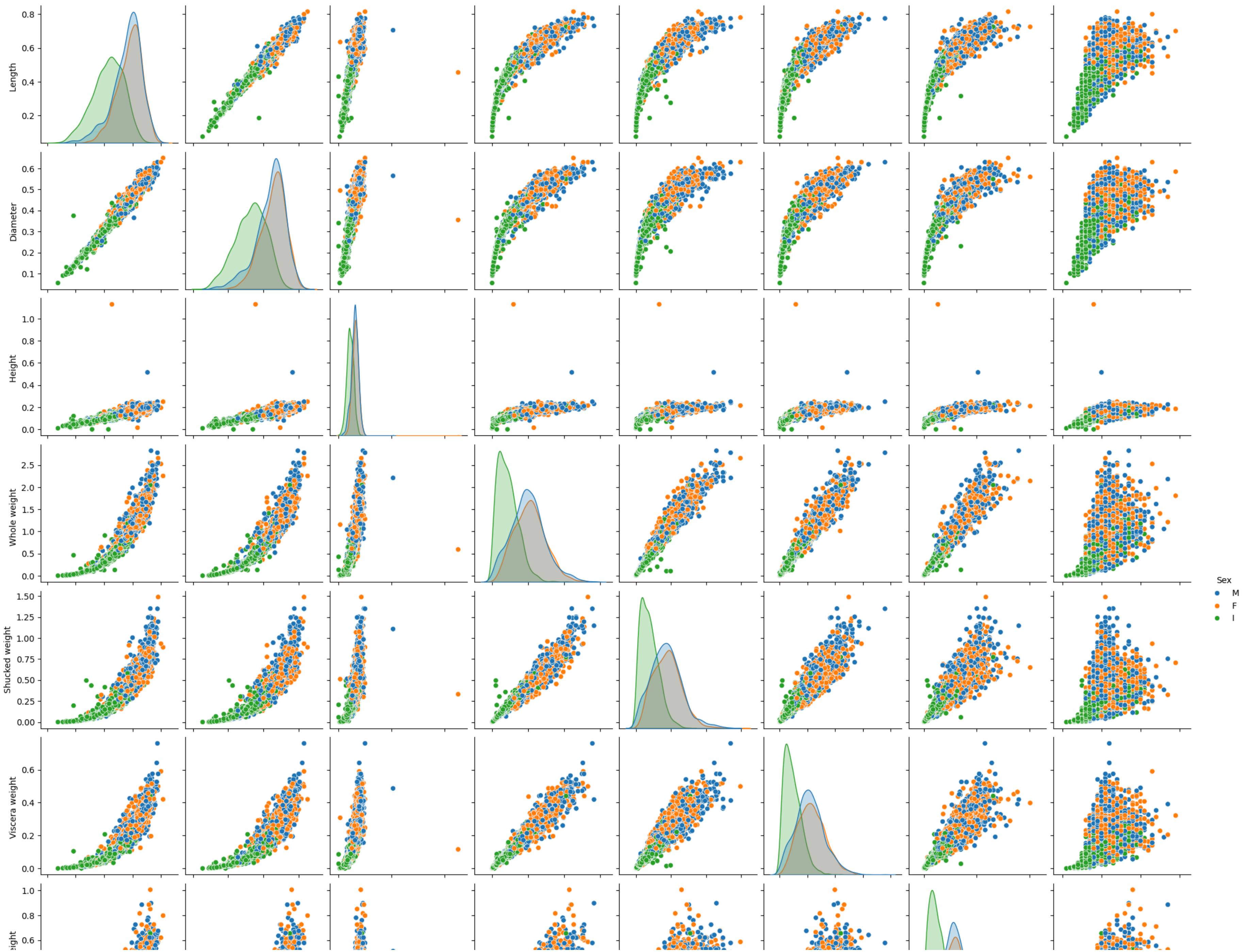
	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7

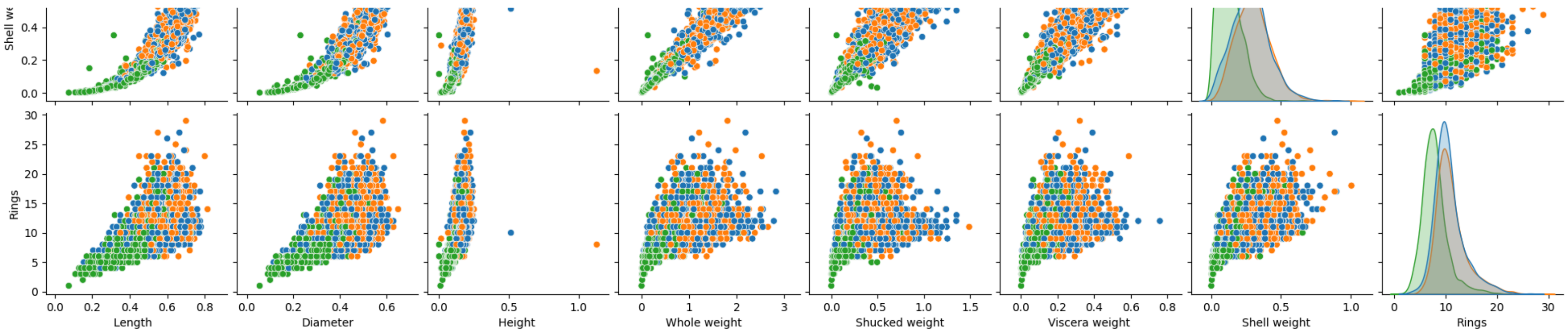
Step 2: Exploratory Data Analysis (EDA)

In [3]:

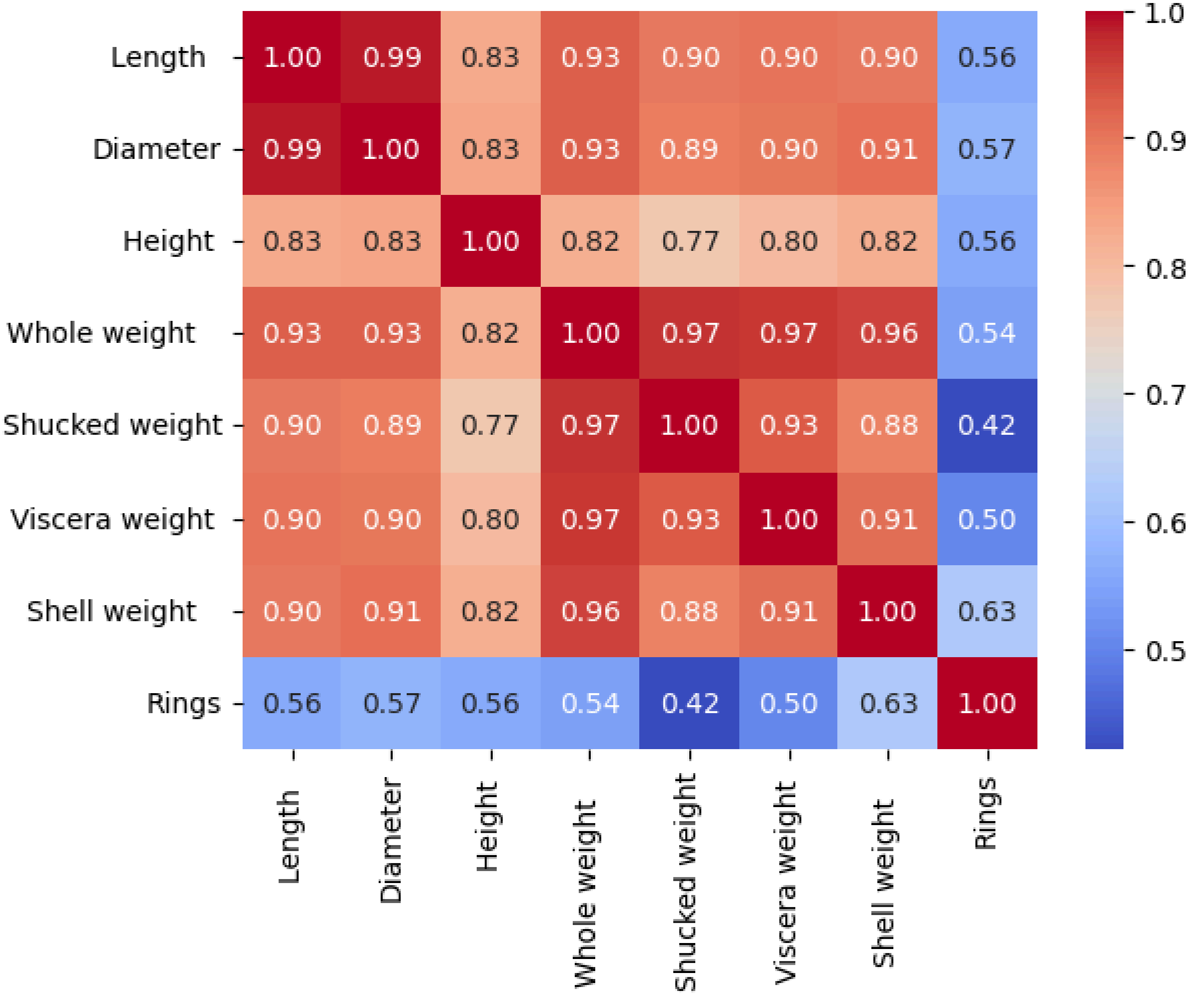
```python
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming abalone_df is your DataFrame
# To display statistical summaries, avoid the 'Sex' column
numeric_data = abalone_df.select_dtypes(include=[float, int])
print(numeric_data.describe())

# If you want to use 'Sex' for the pairplot, it's fine as is
sns.pairplot(abalone_df, hue='Sex')
plt.show()

# For the correlation matrix, exclude 'Sex' or convert it to numeric first
numeric_data = abalone_df.drop(columns=['Sex'])  # Dropping the 'Sex' column
sns.heatmap(numeric_data.corr(), annot=True, cmap='coolwarm', fmt='.2f')
plt.show()
```

```
            Length     Diameter       Height  Whole weight  Shucked weight  \
count  4177.000000  4177.000000  4177.000000   4177.000000     4177.000000
mean      0.523992     0.407881     0.139516      0.828742        0.359367
std       0.120093     0.099240     0.041827      0.490389        0.221963
min       0.075000     0.055000     0.000000      0.002000        0.001000
25%       0.450000     0.350000     0.115000      0.441500        0.186000
50%       0.545000     0.425000     0.140000      0.799500        0.336000
75%       0.615000     0.480000     0.165000      1.153000        0.502000
max       0.815000     0.650000     1.130000      2.825500        1.488000

       Viscera weight  Shell weight        Rings
count     4177.000000   4177.000000  4177.000000
mean         0.180594      0.238831     9.933684
std          0.109614      0.139203     3.224169
min          0.000500      0.001500     1.000000
25%          0.093500      0.130000     8.000000
50%          0.171000      0.234000     9.000000
75%          0.253000      0.329000    11.000000
max          0.760000      1.005000    29.000000
```


Length
Diameter
Height
Whole weight
Shucked weight
Viscera weight
Sex
M
F
I



Shell weight
Rings
Length
Diameter
Height
Whole weight
Shucked weight
Viscera weight
Shell weight
Rings



Step 3: Standardize the Input

In [4]:
```
from sklearn.preprocessing import StandardScaler

# Assume 'Sex' needs to be dropped or transformed into numerical values before scaling
X = abalone_df.drop('Sex', axis=1)
y = abalone_df['Sex']
```

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split the data
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3, random_state=42)
```

Step 4: Build Logistic Regression Model

In [5]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# Initialize and train logistic regression model
log_reg = LogisticRegression(random_state=42)
log_reg.fit(X_train, y_train)

# Predictions
y_pred_log_reg = log_reg.predict(X_test)

# Evaluation
log_reg_accuracy = accuracy_score(y_test, y_pred_log_reg)
log_reg_conf_matrix = confusion_matrix(y_test, y_pred_log_reg)
log_reg_class_report = classification_report(y_test, y_pred_log_reg)

print("Logistic Regression Accuracy:", log_reg_accuracy)
print("Confusion Matrix:\n", log_reg_conf_matrix)
print("Classification Report:\n", log_reg_class_report)
```

```
Logistic Regression Accuracy: 0.569377990430622
Confusion Matrix:
 [[144  69 164]
 [ 20 354  45]
 [142 100 216]]
Classification Report:
               precision    recall  f1-score   support

           F       0.47      0.38      0.42       377
           I       0.68      0.84      0.75       419
           M       0.51      0.47      0.49       458

    accuracy                           0.57      1254
   macro avg       0.55      0.57      0.55      1254
weighted avg       0.55      0.57      0.56      1254
```

Step 6: Build Support Vector Classification Model

In [6]:

```
from sklearn.svm import SVC

# Initialize and train SVC model
svc = SVC(random_state=42)
svc.fit(X_train, y_train)

# Predictions
y_pred_svc = svc.predict(X_test)

# Evaluation
svc_accuracy = accuracy_score(y_test, y_pred_svc)
svc_conf_matrix = confusion_matrix(y_test, y_pred_svc)
svc_class_report = classification_report(y_test, y_pred_svc)

print("Support Vector Classification Accuracy:", svc_accuracy)
print("Confusion Matrix:\n", svc_conf_matrix)
print("Classification Report:\n", svc_class_report)
```

```
Support Vector Classification Accuracy: 0.5606060606060606
Confusion Matrix:
 [[121  55 201]
 [ 17 339  63]
 [124  91 243]]
Classification Report:
               precision    recall  f1-score   support

           F       0.46      0.32      0.38       377
           I       0.70      0.81      0.75       419
           M       0.48      0.53      0.50       458

    accuracy                           0.56      1254
   macro avg       0.55      0.55      0.54      1254
weighted avg       0.55      0.56      0.55      1254
```

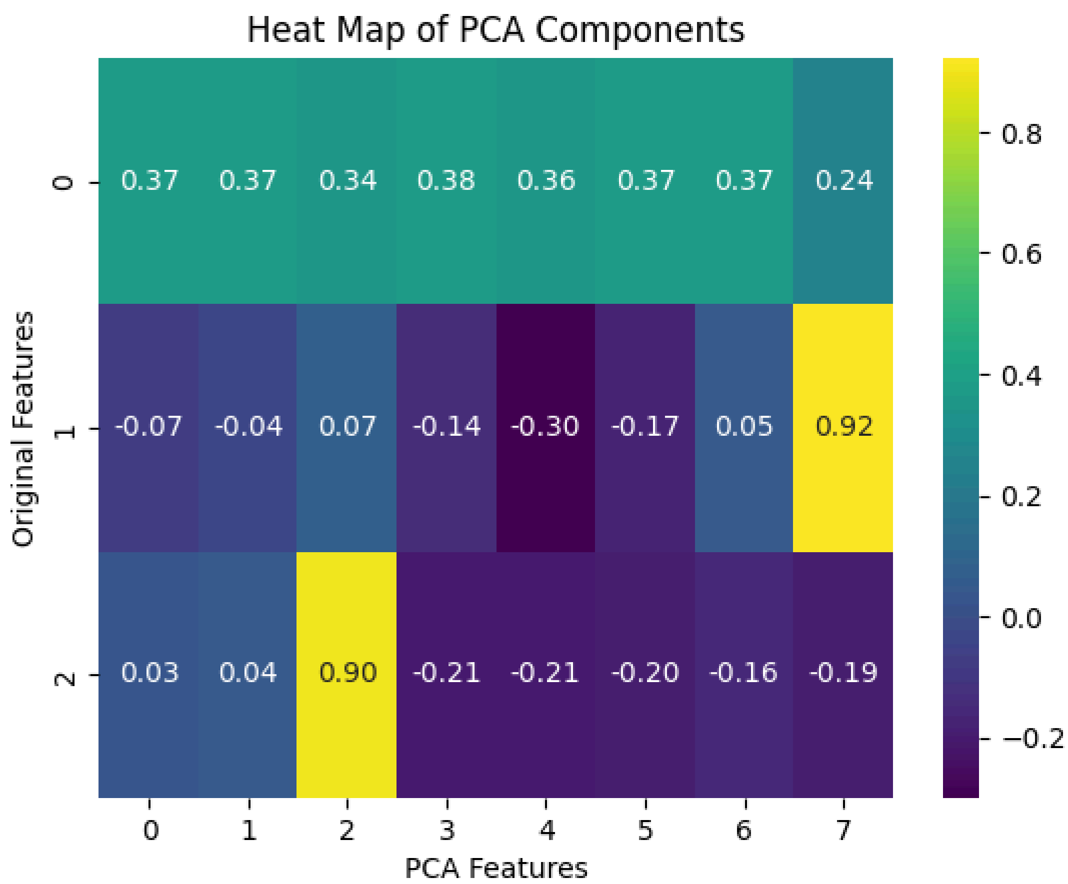
Step 8: Perform PCA

In [7]:
```python
from sklearn.decomposition import PCA

# PCA transformation
pca = PCA(n_components=0.95)  # Adjust the number of components for 95% variance
X_pca = pca.fit_transform(X_scaled)

print("Explained Variance Ratio:", pca.explained_variance_ratio_)
```

```
Explained Variance Ratio: [0.83905489 0.08695162 0.03230539]
```

Step 9: Visualize Coefficients Using a Heat Map

In [8]:
```python
sns.heatmap(pca.components_, annot=True, cmap='viridis', fmt='.2f')
plt.xlabel('PCA Features')
plt.ylabel('Original Features')
plt.title('Heat Map of PCA Components')
plt.show()
```



Steps 10 and 11: Rebuild Models with Principal Components

In [9]:
```
# Split the data transformed by PCA
X_train_pca, X_test_pca, y_train, y_test = train_test_split(X_pca, y, test_size=0.3, random_state=42)

# Rebuild Logistic Regression model
log_reg_pca = LogisticRegression(random_state=42)
log_reg_pca.fit(X_train_pca, y_train)
y_pred_log_reg_pca = log_reg_pca.predict(X_test_pca)

# Rebuild SVC model
```

```
svc_pca = SVC(random_state=42)
svc_pca.fit(X_train_pca, y_train)
y_pred_svc_pca = svc_pca.predict(X_test_pca)

# Evaluation
print("Logistic Regression with PCA Accuracy:", accuracy_score(y_test, y_pred_log_reg_pca))
print("Support Vector Classification with PCA Accuracy:", accuracy_score(y_test, y_pred_svc_pca))
print("Confusion Matrix for Logistic Regression:\n", confusion_matrix(y_test, y_pred_log_reg_pca))
print("Confusion Matrix for SVC:\n", confusion_matrix(y_test, y_pred_svc_pca))
print("Classification Report for Logistic Regression:\n", classification_report(y_test, y_pred_log_reg_pca))
print("Classification Report for SVC:\n", classification_report(y_test, y_pred_svc_pca))
```

```
Logistic Regression with PCA Accuracy: 0.5558213716108453
Support Vector Classification with PCA Accuracy: 0.5661881977671451
Confusion Matrix for Logistic Regression:
 [[102  57 218]
 [ 13 329  77]
 [ 98  94 266]]
Confusion Matrix for SVC:
 [[136  59 182]
 [ 14 332  73]
 [117  99 242]]
Classification Report for Logistic Regression:
               precision    recall  f1-score   support

           F       0.48      0.27      0.35       377
           I       0.69      0.79      0.73       419
           M       0.47      0.58      0.52       458

    accuracy                           0.56      1254
   macro avg       0.55      0.55      0.53      1254
weighted avg       0.55      0.56      0.54      1254

Classification Report for SVC:
               precision    recall  f1-score   support

           F       0.51      0.36      0.42       377
           I       0.68      0.79      0.73       419
           M       0.49      0.53      0.51       458

    accuracy                           0.57      1254
   macro avg       0.56      0.56      0.55      1254
weighted avg       0.56      0.57      0.56      1254
```

In [ ]: