Logistic and SVC Models on Abalone Data

In this notebook, we will execute several data analysis steps on the abalone dataset. We will perform initial data processing, exploratory data analysis, and subsequently, build predictive models using Logistic Regression and Support Vector Machines. Further, we will apply PCA to optimize our models.

Data Acquisition

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

data_path = r'dataset/abalone.csv'
abalone_df = pd.read_csv(data_path)
abalone_df.head()
```

Out[1]:

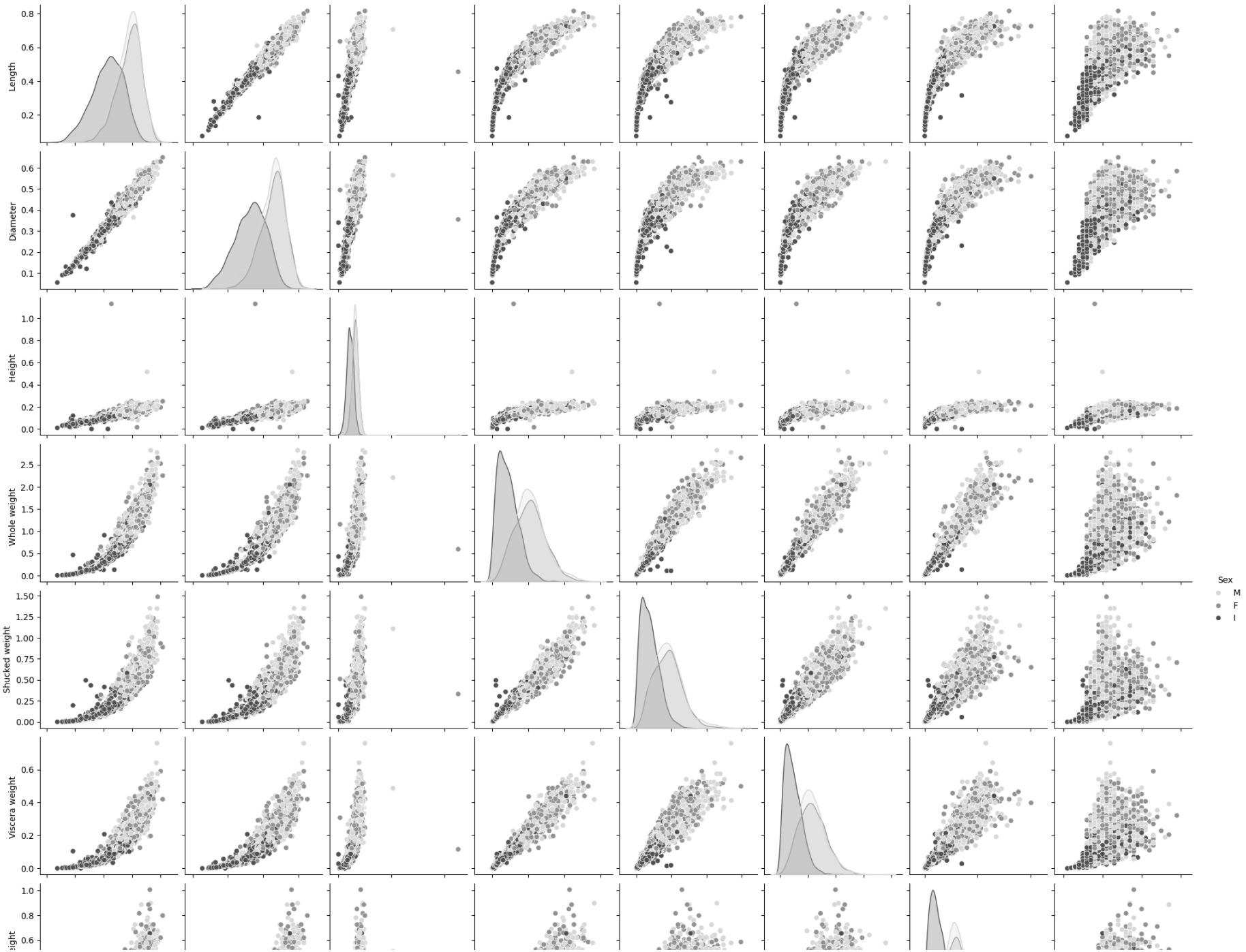
	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10
4	I	0.330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

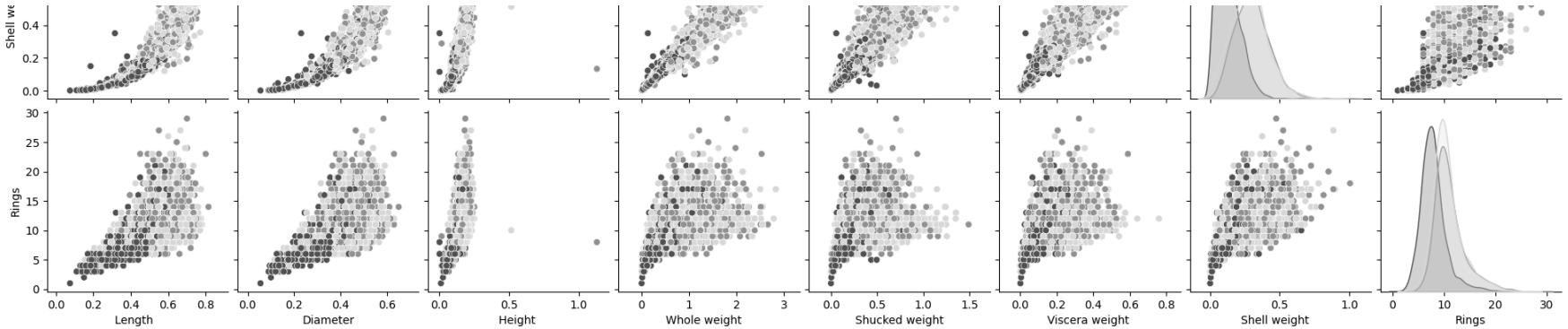
Exploratory Data Analysis (EDA)

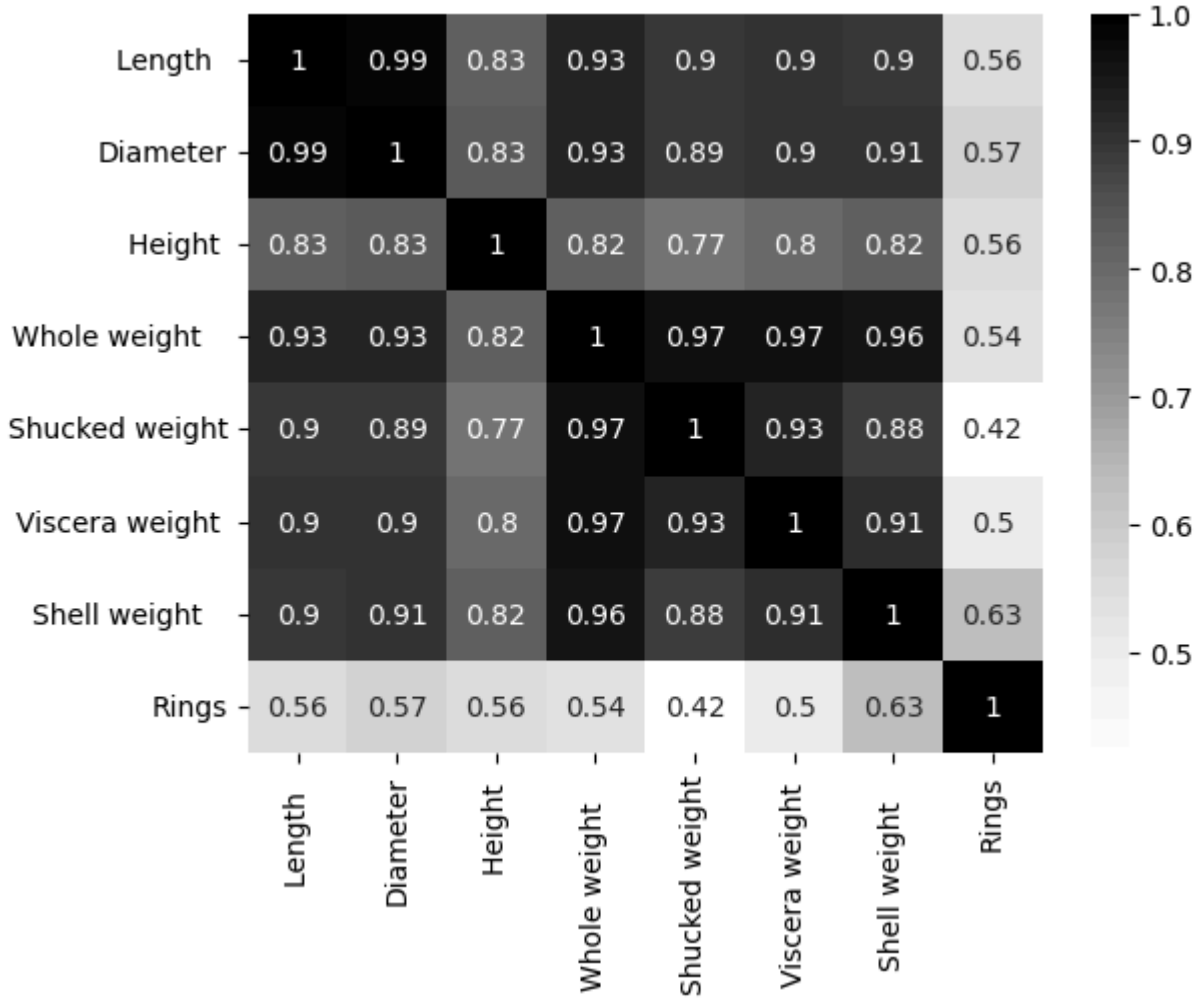
In [19]:
```
import seaborn as sns
import matplotlib.pyplot as plt

sns.pairplot(abalone_df, hue='Sex', palette="Greys")
plt.show()
```

```
correlation_matrix = abalone_df.select_dtypes(include=[float, int]).corr()
sns.heatmap(correlation_matrix, annot=True, cmap='Greys')
plt.show()
```


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


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



Data Standardization

In [9]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score

X = abalone_df.drop(['Sex'], axis=1)
y = abalone_df['Sex']
```

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

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3, random_state=42)
```

Logistic Regression Model

In [10]:
```
from sklearn.linear_model import LogisticRegression
log_reg = LogisticRegression(random_state=42)
log_reg.fit(X_train, y_train)

y_pred_log_reg = log_reg.predict(X_test)

print(f'Accuracy: {accuracy_score(y_test, y_pred_log_reg)}')
print('Confusion Matrix:')
print(confusion_matrix(y_test, y_pred_log_reg))
```

```
Accuracy: 0.569377990430622
Confusion Matrix:
[[144  69 164]
 [ 20 354  45]
 [142 100 216]]
```

Support Vector Machine (SVM) Model

In [11]:
```
from sklearn.svm import SVC
svc_model = SVC(random_state=42)
svc_model.fit(X_train, y_train)

y_pred_svc = svc_model.predict(X_test)

print(f'Accuracy: {accuracy_score(y_test, y_pred_svc)}')
print('Confusion Matrix:')
print(confusion_matrix(y_test, y_pred_svc))
```

```
Accuracy: 0.5606060606060606
Confusion Matrix:
[[121  55 201]
 [ 17 339  63]
 [124  91 243]]
```

Principal Component Analysis (PCA)

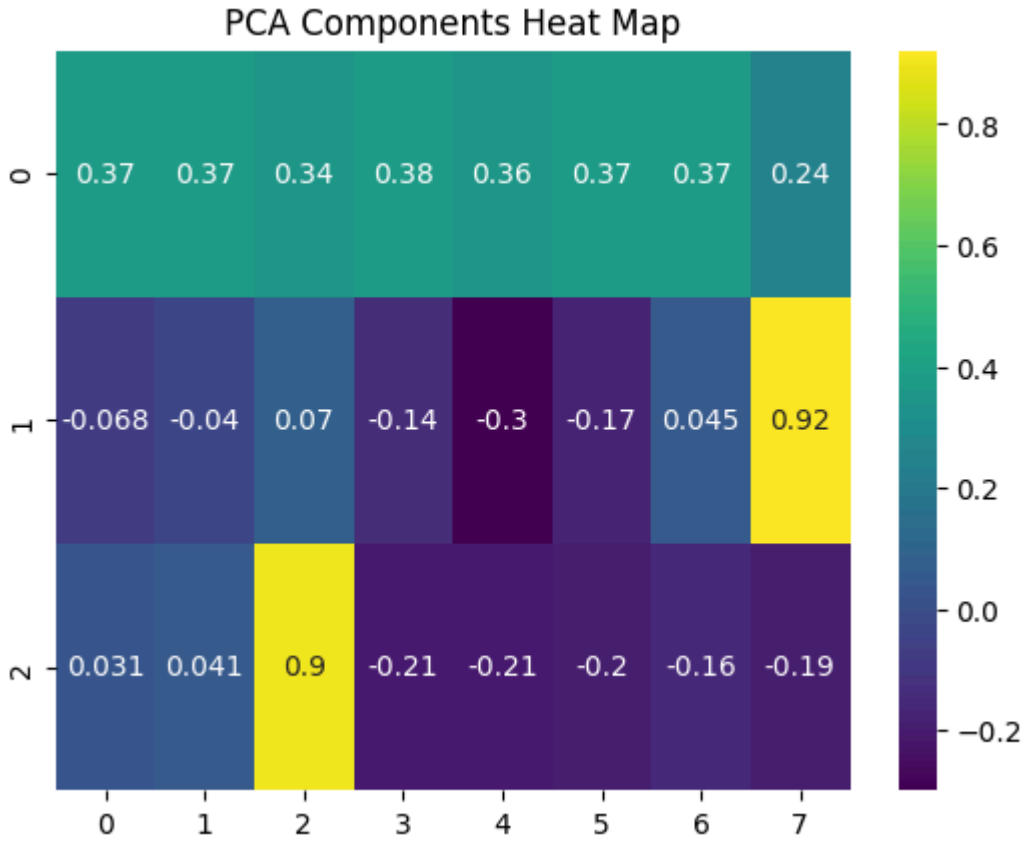
In [12]:
```
from sklearn.decomposition import PCA
pca = PCA(n_components=0.95)
X_pca = pca.fit_transform(X_scaled)

print('PCA Components Explained Variance Ratio:')
print(pca.explained_variance_ratio_)
```

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

Visualization of PCA Components

In [13]:
```
sns.heatmap(pca.components_, annot=True, cmap='viridis')
plt.title('PCA Components Heat Map')
plt.show()
```



Model Rebuilding with PCA

In [14]:
```
X_train_pca, X_test_pca, y_train, y_test = train_test_split(X_pca, y, test_size=0.3, random_state=42)
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)

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

```
print('PCA-based Logistic Regression Accuracy:', accuracy_score(y_test, y_pred_log_reg_pca))
print('PCA-based SVC Accuracy:', accuracy_score(y_test, y_pred_svc_pca))
```

```
PCA-based Logistic Regression Accuracy: 0.5558213716108453
PCA-based SVC Accuracy: 0.5661881977671451
```