In [36]:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, accuracy_score
```

Load the dataset

In [2]:

```
data = pd.read_csv('wisconsin.csv')
```

Data preprocessing

Encode 'Class' column to numerical values (benign: 0, malignant: 1)

In [3]:

```
label_encoder = LabelEncoder()
data['Class'] = label_encoder.fit_transform(data['Class'])
```

Split the dataset into features (X) and target (y)

In [8]:

```
X = data.drop('Class', axis=1)
y = data['Class']
```

Split the data into training and testing sets

In [9]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Create a LogisticRegression model

In [37]:

```
clf = LogisticRegression(random_state=42)
```

Train the model

In [38]:

```
X = data.drop('Class', axis=1)
y = data['Class']
```

Make predictions

In [39]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [40]:

```
clf = LogisticRegression(random_state=42)
```

In [41]:

```
clf.fit(X_train, y_train)  # Call 'fit' to train the model
```

Out[41]:

```
LogisticRegression(random_state=42)
```

Make predictions

In [42]:

```
y_pred = clf.predict(X_test)  # Now you can make predictions
```

Evaluate the model

In [43]:

```
accuracy = accuracy_score(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred, target_names=['benign', 'malig
```

In [44]:

```
print(f"Accuracy: {accuracy}")
print("Classification Report:")
print(classification_rep)
```

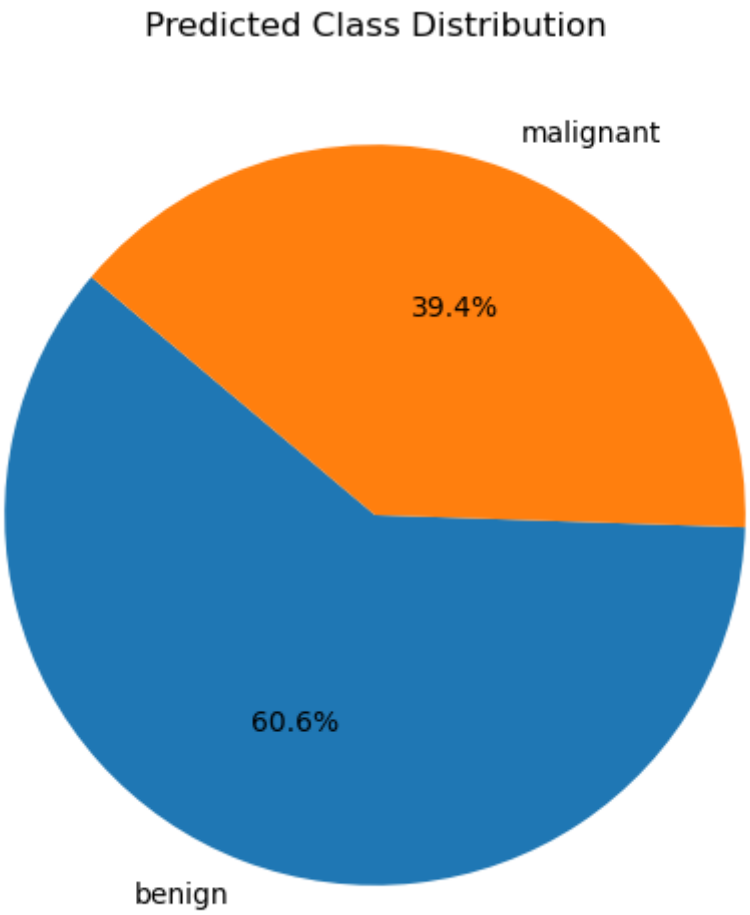
```
Accuracy: 0.9562043795620438
Classification Report:
              precision    recall  f1-score   support

      benign       0.94      0.99      0.96        79
   malignant       0.98      0.91      0.95        58

    accuracy                           0.96       137
   macro avg       0.96      0.95      0.95       137
weighted avg       0.96      0.96      0.96       137
```

In [45]:

```
class_counts = np.bincount(y_pred)
class_labels = ['benign', 'malignant']
plt.figure(figsize=(6, 6))
plt.pie(class_counts, labels=class_labels, autopct='%1.1f%%', startangle=140)
plt.title('Predicted Class Distribution')
plt.show()
```



In [ ]: