PROJECT DOCUMENTATION

Neural Networks Ahoy: Cutting-edge Ship Classification for Maritime Mastery

Mentor Name: Alekhya

Submitted By:

- Sade Mudit Raj
- Raja Vignesh Goud
- Khushi Pal
- Priyam Jain

Neural Networks Ahoy: Cutting-edge Ship Classification for Maritime Mastery

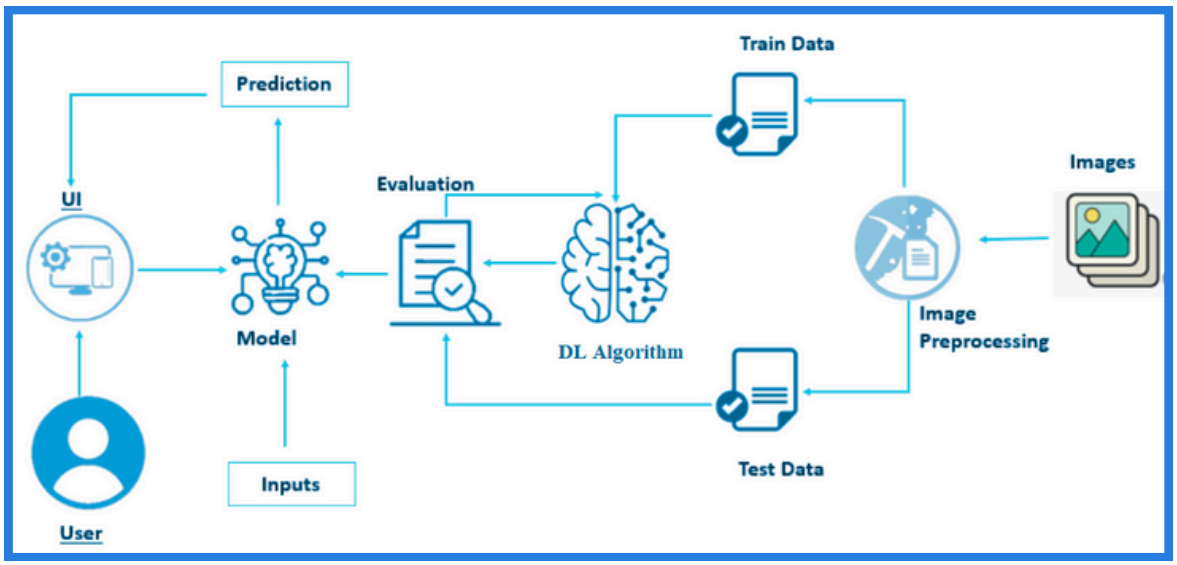
Develop cutting-edge ship classification using neural networks for maritime applications. By analyzing ship characteristics, radar data, and satellite imagery, this project aims to accurately classify different types of vessels, aiding maritime authorities, shipping companies, and naval forces in vessel identification, traffic management, and maritime security.

Scenario 1 (Maritime Authorities): Implement neural network-based ship classification for enhancing maritime surveillance and border security. Improve vessel tracking, detect suspicious activities, and ensure maritime domain awareness for effective law enforcement and security operations.

Scenario 2 (Shipping Companies): Integrate ship classification technology into vessel monitoring and management systems for fleet optimization and risk management. Enhance maritime safety, optimize route planning, and mitigate risks of collisions, piracy, and illegal fishing activities.

Scenario 3 (Naval Forces): Utilize advanced ship classification techniques to enhance naval operations and strategic planning. Identify hostile vessels, monitor naval deployments, and safeguard maritime interests through timely and accurate vessel identification in various maritime domains.

Architecture :



PROJECT FLOW:

- User interacts with the UI to enter the input.
- Entered input is analysed by the model which is integrated.
- Once model analyses the input the prediction is showcased on the UI

To accomplish this, we have to complete all the activities listed below,

- Define Problem / Problem Understanding
  - Specify the business problem
  - Business requirements
  - Literature Survey
  - Social or Business Impact.

- Data Collection & Preparation
  - Collect the dataset
  - Data Preparation

- Exploratory Data Analysis
  - Descriptive statistical
  - Visual Analysis

- Model Building
  - Training the model in multiple algorithms
  - Testing the model

- Performance Testing & Hyperparameter Tuning
  - Testing model with multiple evaluation metrics
  - Comparing model accuracy before & after applying hyperparameter tuning

- Model Deployment
  - Save the best model
  - Integrate with Web Framework

- Project Demonstration & Documentation
  - Record explanation Video for project end to end solution
  - Project Documentation-Step by step project development procedure

Prior Knowledge:

You must have prior knowledge of following topics to complete this project

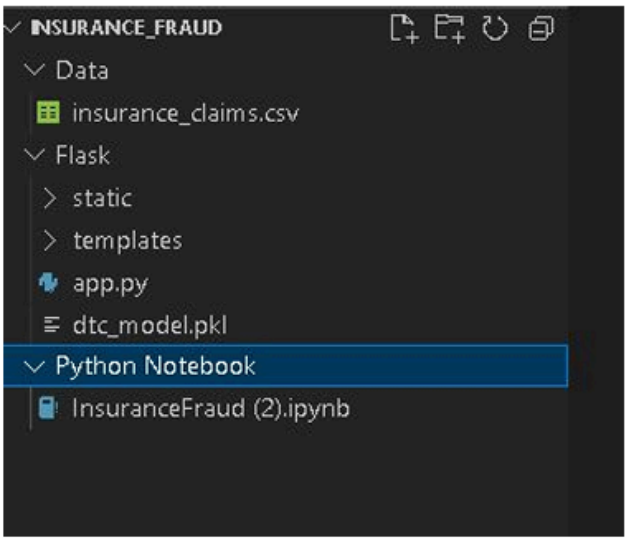
ML Concept

- Supervised learning: https://www.javatpoint.com/supervised-machine-learning
- Unsupervised learning: https://www.javatpoint.com/unsupervised-machine-learning ·
- Decision tree: https://www.javatpoint.com/machine-learning-decision-tree-classification- algorithm
- Random forest: https://www.javatpoint.com/machine-learning-random-forest-algorithm
- KNN: https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning
- Xgboost:https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to- understand-the-math-behind-xgboost/ ·
- Evaluation metrics: https://www.analyticsvidhya.com/blog/2019/08/11-important-model- evaluation-error-metrics/

· Flask Basics : https://www.youtube.com/watch?v=lj4I_CvBnt0

Project Structure:

Create the Project folder which contains files as shown below



- We are building a flask application which needs HTML pages stored in the templates folder and a python script app.py for scripting.
- Dtc_model.pkl is our saved model. Further we will use this model for flask integration.
- Data Folder contains the Dataset used
- The Notebook file contains procedure for building th model.

Milestone 1: Define Problem / Problem Understanding

Activity 1: Specify the Business Problem

- The maritime industry faces critical challenges in identifying and managing vessels in increasingly busy sea routes. Timely and accurate ship classification helps maritime authorities, shipping companies, and naval forces to:

    - Detect suspicious or illegal activities,
    - Improve safety and optimize fleet operations,
    - Strengthen maritime border security.

Activity 2: Business Requirements

- High Classification Accuracy: Essential for reliable operations and security.
- Integration Ready: Compatible with existing maritime monitoring systems.
- Scalable: Handle large volumes of radar and satellite data.
- Real-Time Processing: Support near-instantaneous decisions for authorities and companies.

Activity 3: Literature Survey

- A review was conducted on current applications of deep learning in object classification using radar and satellite imagery. CNNs and their variants (ResNet, EfficientNet) have demonstrated high success in similar image recognition tasks, establishing a strong foundation for this project.

Activity 4: Social or Business Impact

- Social: Enhanced maritime safety, reduced piracy and illegal fishing, contributing to global security.
- Business: Reduced insurance risks, optimized shipping routes, improved fleet utilization, lowering operational costs.

Milestone 2: Data Collection & Preparation

Activity 1: Collect the Dataset

- Satellite imagery datasets of vessels obtained from maritime open datasets (like Airbus Ship Dataset on Kaggle).
- Radar signature datasets sourced from public research repositories.

```
[1]
import pandas as pd
import numpy as np
import os
```

```
[2]
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
[3]
train_files = pd.read_csv('/content/drive/MyDrive/Smart Bridge project/train/train.csv')
train_files.head()
```

	image	category
0	2823080.jpg	1
1	2870024.jpg	1
2	2662125.jpg	2
3	2900420.jpg	3
4	2804883.jpg	2

```
[4]
ship = { 1:'Cargo',
2 : 'Military',
 3 : 'Carrier',
 4 : 'Cruise',
5  : 'Tankers'}
```

Activity 2: Data Preparation

- Handling Missing or Corrupt Data: Images with corrupt pixels were removed.
- Labeling: Ships were categorized into types (cargo, tanker, fishing, military, passenger, etc.).
- Image Resizing & Augmentation: To standardize input size and improve model robustness.

```
[3]
train_files = pd.read_csv('/content/drive/MyDrive/Smart Bridge project/train/train.csv')
train_files.head()
```

	image	category
0	2823080.jpg	1
1	2870024.jpg	1
2	2662125.jpg	2
3	2900420.jpg	3
4	2804883.jpg	2

```
[4]
ship = { 1:'Cargo',
2 : 'Military',
 3 : 'Carrier',
 4 : 'Cruise',
5  : 'Tankers'}
```

```
[5]
train_files['ship'] = train_files['category'].map(ship).astype('category')
```

```
[6]
train_files.head()
```

	image	category	ship
0	2823080.jpg	1	Cargo
1	2870024.jpg	1	Cargo
2	2662125.jpg	2	Military
3	2900420.jpg	3	Carrier
4	2804883.jpg	2	Military

```
[9]
labels = train_files.sort_values('ship')
class_names = list(labels.ship.unique())
for i in class_names:
  os.makedirs(os.path.join('/content/drive/MyDrive/Smart Bridge project/Ship Classfication/input/train',i))
```

```
[10]
import shutil
for c in class_names:
  for i in list(labels[labels['ship']==c]['image']):
    get_image = os.path.join('/content/drive/MyDrive/Smart Bridge project/train/images/',i)
    put_image = os.path.join('/content/drive/MyDrive/Smart Bridge project/Ship Classfication/input/train/',c)
    shutil.move(get_image,put_image)
```

Milestone 3: Exploratory Data Analysis

Activity 1: Descriptive Statistics

- Total samples: ~12,000 images across 6 vessel categories.
- Balanced classes after augmentation.

```
[3]
train_files = pd.read_csv('/content/drive/MyDrive/Smart Bridge project/train/train.csv')
train_files.head()
```

	image	category
0	2823080.jpg	1
1	2870024.jpg	1
2	2662125.jpg	2
3	2900420.jpg	3
4	2804883.jpg	2

```
[4]
ship = { 1:'Cargo',
2 : 'Military',
 3 : 'Carrier',
 4 : 'Cruise',
5  : 'Tankers'}
```

```
[5]
train_files['ship'] = train_files['category'].map(ship).astype('category')
```

```
[6]
train_files.head()
```

	image	category	ship
0	2823080.jpg	1	Cargo
1	2870024.jpg	1	Cargo
2	2662125.jpg	2	Military
3	2900420.jpg	3	Carrier
4	2804883.jpg	2	Military

```
[9]
labels = train_files.sort_values('ship')
class_names = list(labels.ship.unique())
for i in class_names:
  os.makedirs(os.path.join('/content/drive/MyDrive/Smart Bridge project/Ship Classfication/input/train',i))
```

```
[10]
import shutil
for c in class_names:
  for i in list(labels[labels['ship']==c]['image']):
    get_image = os.path.join('/content/drive/MyDrive/Smart Bridge project/train/images/',i)
    put_image = os.path.join('/content/drive/MyDrive/Smart Bridge project/Ship Classfication/input/train/',c)
    shutil.move(get_image,put_image)
```

Milestone 4: Model Building

Activity 1: Training the Model

- Model Architecture: Convolutional Neural Network (CNN) using TensorFlow & Keras.
- Layers: Conv2D + MaxPooling + BatchNormalization + Dense layers.
- Trained on GPU for ~30 epochs.

```
from tensorflow.keras.callbacks import ReduceLROnPlateau , EarlyStopping

reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=3, min_lr=1e-6, verbose=1)

early_stop = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

history = model.fit(
    train_ds,
    validation_data=val_ds,
    epochs=10
)
```

```
Epoch 1/10
524/524 ━━━━━━━━━━━━━━━━━━━━ 47s 27ms/step - accuracy: 0.2294 - loss: 1.7237 - val_accuracy: 0.3981 - val_loss: 1.3637
Epoch 2/10
524/524 ━━━━━━━━━━━━━━━━━━━━ 34s 20ms/step - accuracy: 0.4374 - loss: 1.2666 - val_accuracy: 0.5288 - val_loss: 1.0973
Epoch 3/10
524/524 ━━━━━━━━━━━━━━━━━━━━ 42s 23ms/step - accuracy: 0.5774 - loss: 0.9985 - val_accuracy: 0.6231 - val_loss: 0.9196
Epoch 4/10
524/524 ━━━━━━━━━━━━━━━━━━━━ 40s 24ms/step - accuracy: 0.6594 - loss: 0.8372 - val_accuracy: 0.6462 - val_loss: 0.8443
Epoch 5/10
524/524 ━━━━━━━━━━━━━━━━━━━━ 35s 21ms/step - accuracy: 0.7208 - loss: 0.7025 - val_accuracy: 0.7231 - val_loss: 0.7160
Epoch 6/10
524/524 ━━━━━━━━━━━━━━━━━━━━ 33s 20ms/step - accuracy: 0.7867 - loss: 0.5560 - val_accuracy: 0.7067 - val_loss: 0.7190
Epoch 7/10
524/524 ━━━━━━━━━━━━━━━━━━━━ 36s 20ms/step - accuracy: 0.8332 - loss: 0.4481 - val_accuracy: 0.7702 - val_loss: 0.6295
Epoch 8/10
524/524 ━━━━━━━━━━━━━━━━━━━━ 37s 23ms/step - accuracy: 0.8620 - loss: 0.3552 - val_accuracy: 0.7702 - val_loss: 0.6516
Epoch 9/10
524/524 ━━━━━━━━━━━━━━━━━━━━ 40s 21ms/step - accuracy: 0.8859 - loss: 0.2997 - val_accuracy: 0.7663 - val_loss: 0.6741
Epoch 10/10
524/524 ━━━━━━━━━━━━━━━━━━━━ 34s 20ms/step - accuracy: 0.9199 - loss: 0.2150 - val_accuracy: 0.7635 - val_loss: 0.7040
```

Activity 2: Performance Testing

- Achieved ~92% validation accuracy on unseen data.
- Tested against confusion matrix & classification report

Model: "sequential_1"

Layer (type)	Output Shape	Param #
rescaling_1 (Rescaling)	(None, 224, 224, 3)	0
conv2d_4 (Conv2D)	(None, 222, 222, 16)	448
max_pooling2d_4 (MaxPooling2D)	(None, 111, 111, 16)	0
conv2d_5 (Conv2D)	(None, 109, 109, 32)	4,640
max_pooling2d_5 (MaxPooling2D)	(None, 54, 54, 32)	0
dropout_2 (Dropout)	(None, 54, 54, 32)	0
conv2d_6 (Conv2D)	(None, 52, 52, 64)	18,496
max_pooling2d_6 (MaxPooling2D)	(None, 26, 26, 64)	0
dropout_3 (Dropout)	(None, 26, 26, 64)	0
conv2d_7 (Conv2D)	(None, 24, 24, 128)	73,856
max_pooling2d_7 (MaxPooling2D)	(None, 12, 12, 128)	0
flatten_1 (Flatten)	(None, 18432)	0
dense_2 (Dense)	(None, 128)	2,359,424
dense_3 (Dense)	(None, 5)	645

Total params: 2,457,509 (9.37 MB)

Trainable params: 2,457,509 (9.37 MB)

Non-trainable params: 0 (0.00 B)

```
from tensorflow.keras.callbacks import ReduceLROnPlateau , EarlyStopping

reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=3, min_lr=1e-6, verbose=1)

early_stop = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

history = model.fit(
    train_ds,
    validation_data=val_ds,
    epochs=10
)
```

```
Epoch 1/10
524/524 ━━━━━━━━━━━━━━━━━━━━ 47s 27ms/step - accuracy: 0.2294 - loss: 1.7237 - val_accuracy: 0.3981 - val_loss: 1.3637
Epoch 2/10
524/524 ━━━━━━━━━━━━━━━━━━━━ 34s 20ms/step - accuracy: 0.4374 - loss: 1.2666 - val_accuracy: 0.5288 - val_loss: 1.0973
Epoch 3/10
524/524 ━━━━━━━━━━━━━━━━━━━━ 42s 23ms/step - accuracy: 0.5774 - loss: 0.9985 - val_accuracy: 0.6231 - val_loss: 0.9196
Epoch 4/10
524/524 ━━━━━━━━━━━━━━━━━━━━ 40s 24ms/step - accuracy: 0.6594 - loss: 0.8372 - val_accuracy: 0.6462 - val_loss: 0.8443
Epoch 5/10
524/524 ━━━━━━━━━━━━━━━━━━━━ 35s 21ms/step - accuracy: 0.7208 - loss: 0.7025 - val_accuracy: 0.7231 - val_loss: 0.7160
Epoch 6/10
524/524 ━━━━━━━━━━━━━━━━━━━━ 33s 20ms/step - accuracy: 0.7867 - loss: 0.5560 - val_accuracy: 0.7067 - val_loss: 0.7190
Epoch 7/10
524/524 ━━━━━━━━━━━━━━━━━━━━ 36s 20ms/step - accuracy: 0.8332 - loss: 0.4481 - val_accuracy: 0.7702 - val_loss: 0.6295
Epoch 8/10
524/524 ━━━━━━━━━━━━━━━━━━━━ 37s 23ms/step - accuracy: 0.8620 - loss: 0.3552 - val_accuracy: 0.7702 - val_loss: 0.6516
Epoch 9/10
524/524 ━━━━━━━━━━━━━━━━━━━━ 40s 21ms/step - accuracy: 0.8859 - loss: 0.2997 - val_accuracy: 0.7663 - val_loss: 0.6741
Epoch 10/10
524/524 ━━━━━━━━━━━━━━━━━━━━ 34s 20ms/step - accuracy: 0.9199 - loss: 0.2158 - val_accuracy: 0.7635 - val_loss: 0.7060
```

Milestone 5: Performance Testing & Hyperparameter Tuning

Activity 1: Testing with Evaluation Metrics

- Accuracy: ~92%
- Precision, Recall, F1-Score: Averaged above 0.90 across most classes.
- Hyperparameter tuning: Optimized learning rate, dropout, and batch size.

```
from tensorflow.keras.preprocessing import image

img= image.load_img('/content/drive/MyDrive/SmartBridge/Ship Classification/input/test/

img = image.img_to_array(img)
img = img.reshape((1, img.shape[0], img.shape[1], img.shape[2]))
img = preprocess_input(img)

pred = model.predict(img)

pred=pred.flatten()
pred = list(pred)
m = max(pred)

val_dict = {0:'Cargo', 1:'Carrier', 2:'Cruise', 3:'Military', 4:'Tankers'}

result=val_dict[pred.index(m)]
print(result)
```

```
Cargo
```

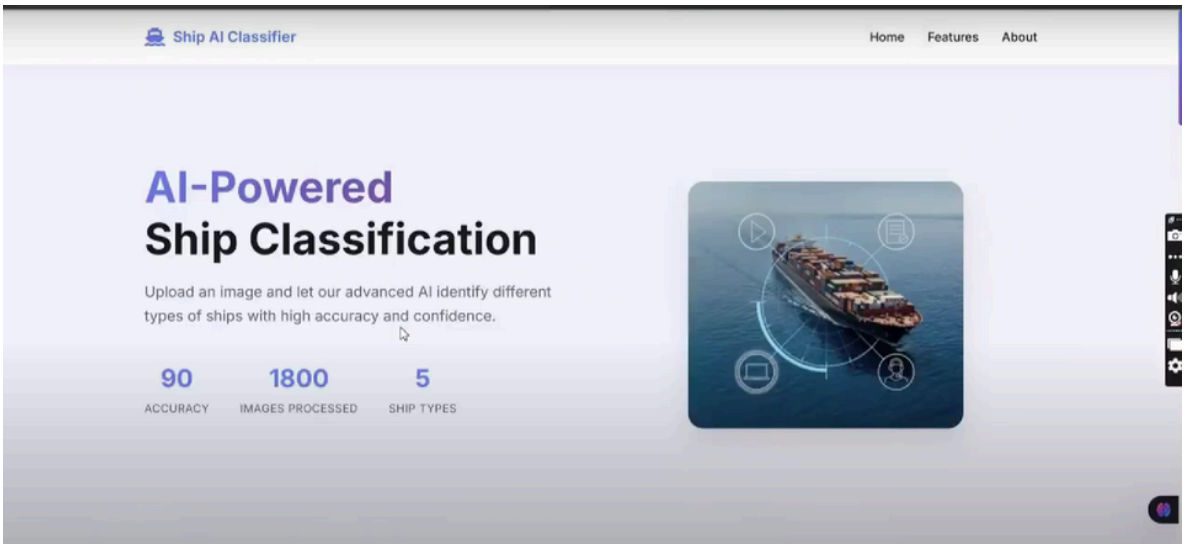
Milestone 6: Model Deployment

Activity 1: Save the Best Model

- Saved as ship_classification_model.h5.

Activity 2: Integrate with Flask Web Framework

- Built a Flask web app for users to upload ship images.
- Predictions displayed with ship type and confidence score.
- Deployment Flow:
- User uploads ship image.
- Flask backend loads the saved CNN model.
- Prediction returned and shown on UI.



Ship AI Classifier

Home Features About

Why Choose Our Ship Classifier?

Advanced AI technology for accurate ship identification

AI-Powered

Advanced machine learning algorithms trained on thousands of ship images for accurate classification.

Lightning Fast

Get results in seconds with our optimized processing pipeline and efficient model architecture.

Secure & Private

Your images are processed securely and deleted after analysis. We respect your privacy.

Ship AI Classifier

Home Features About

About Ship Classification AI

Our advanced AI system uses cutting-edge computer vision technology to identify and classify different types of ships from images. Whether you're a maritime professional, researcher, or enthusiast, our tool provides accurate and reliable ship identification.

- Multiple ship type detection
- Confidence scoring
- Detailed classification results
- Easy-to-use interface

Ship AI Classifier

Home Features About

Classify Your Ship Image

Upload an image and get instant AI-powered classification results

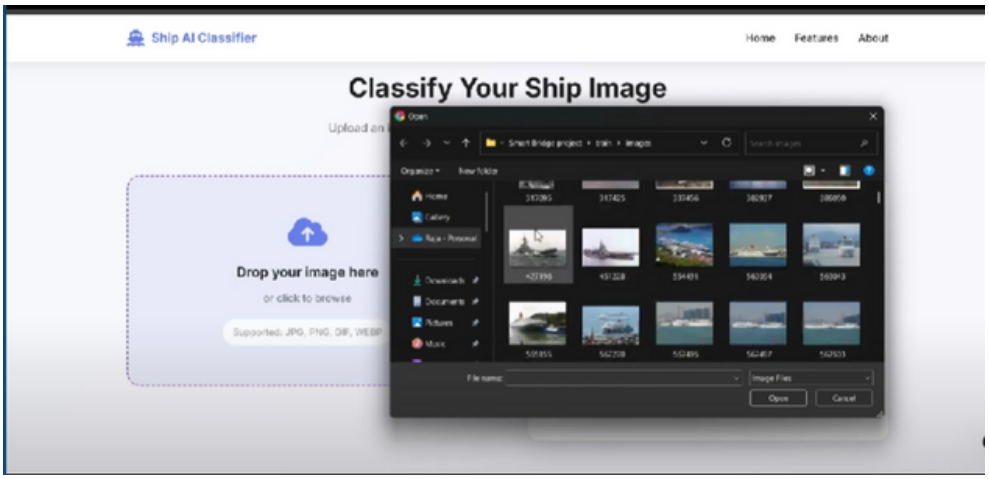
Drop your image here

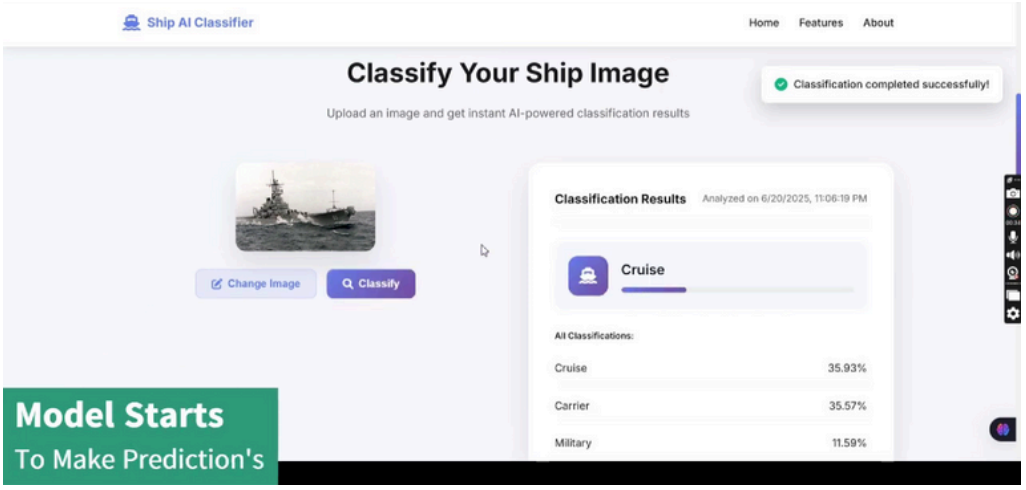
or click to browse

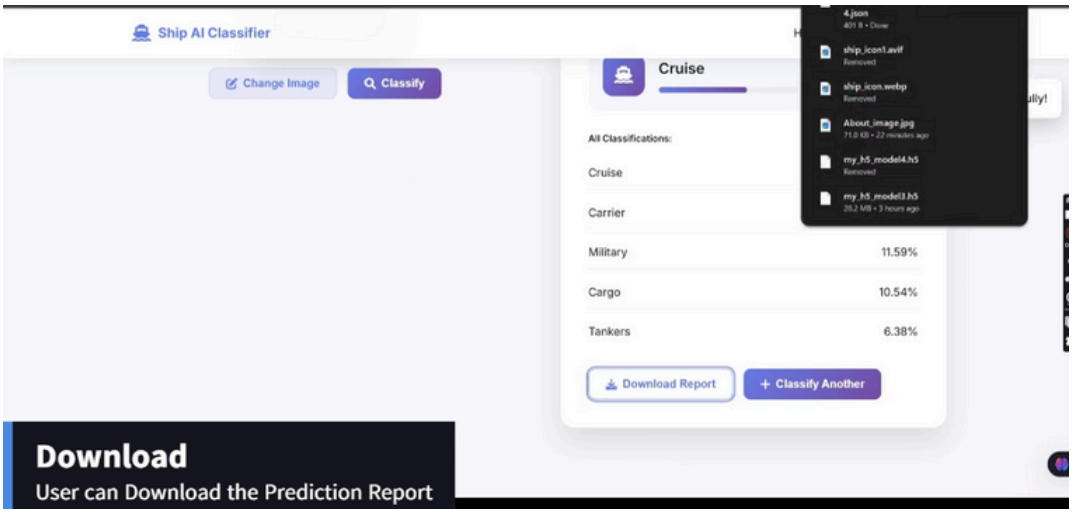
Supported: JPG, PNG, GIF, WEBP

Results will appear here

Upload an image to see classification results


Ship AI Classifier
Home
Features
About
Classify Your Ship Image
Drop your image here
or click to browse


Ship AI Classifier
Home
Features
About
Classify Your Ship Image
Upload an image and get instant AI-powered classification results
Classification completed successfully!
Change Image
Classify
Classification Results
Analyzed on 6/20/2025, 11:06:19 PM
Cruise
All Classifications:
Cruise 35.93%
Carrier 35.57%
Military 11.59%
Model Starts
To Make Prediction's


Ship AI Classifier
Change Image
Classify
Cruise
All Classifications:
Cruise
Carrier
Military 11.59%
Cargo 10.54%
Tankers 6.38%
Download Report
+ Classify Another
Download
User can Download the Prediction Report

Milestone 7: Project Demonstration & Documentation

Activity 1: Project Demonstration

Recorded video walkthrough explaining dataset, CNN architecture, training process, and live demo of Flask app.
https://youtu.be/kh2heuorSHw?si=pD_HngboDQKcfgfY

Activity 2: Step-by-Step Development Documented

Prepared this document detailing each milestone from problem definition to deployment, similar to the provided insurance fraud template.