Internship Final Report: AI & ML Tasks
Submitted by: Sameer Gandhi
Domain: Artificial Intelligence & Machine Learning
Internship Duration: 27th October 2025 - 27th November 2025

Table of Contents

---

1. Introduction

This report contains the final submissions of two selected internship tasks: Spam Mail Detector and House Price Prediction. The objective of these tasks is to demonstrate the practical application of AI & ML concepts in real-world problems.

---

2. Objectives

Task 1: Spam Mail Detector

- Classify emails as spam or non-spam (ham)
- Apply text preprocessing and feature extraction
- Evaluate using accuracy, precision, and F1-score

Task 2: House Price Prediction

- Predict house prices based on numerical features
- Apply regression modeling

- Evaluate using metrics like MSE and R^2

---

3. Dataset Description

Spam Mail Detector:

- Dataset: SMS Spam Collection (UCI) or Enron Email Dataset
- Features: Message text, Label (spam/ham)

House Price Prediction:

- Dataset: California Housing Dataset
- Rows: 20,640 | Features: 8 numerical features
- Features: MedInc, HouseAge, AveRooms, AveBedrms, Population, AveOccup, Latitude, Longitude

---

4. Methodology

Spam Mail Detector:

- Load dataset and labels
- Preprocess text: lowercase, remove stopwords, tokenization
- Convert text to numeric features using Bag of Words or TF-IDF
- Split into training and testing sets
- Train Naive Bayes or Logistic Regression model
- Evaluate performance using accuracy, precision, and F1-score

House Price Prediction:

- Load and explore dataset distributions
- Handle missing values and normalize features
- Split into training and testing sets
- Train Linear Regression model
- Evaluate using Mean Squared Error (MSE) and R^2 score

---

5. Code Implementation

Spam Mail Detector:

```
[#  Complete  Spam Detector with Interactive Mode
# Created for Sameer gandhi

import os
import json
import joblib
import argparse
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import train_test_split

MODEL_FILE = "spam_model.pkl"
VEC_FILE = "vectorizer.pkl"
DATA_FILE = "spam_data.json"

# ---------------------------
# Load or Create Dataset
# ---------------------------
def load_dataset():
    if not os.path.exists(DATA_FILE):
        data = {
            "messages": [
                ("Congratulations! You won ₹50000 lottery", "spam"),
                ("Get free recharge now!!!", "spam"),
                ("Hello Sameer, your meeting is at 5 PM", "ham"),
                ("Your package has been shipped", "ham")
            ]
        }
        with open(DATA_FILE, "w") as f:
            json.dump(data, f, indent=4)
    else:
        with open(DATA_FILE, "r") as f:
            data = json.load(f)
    return data["messages"]

# ---------------------------
# Train the Model
# ---------------------------
def train_model():
    dataset = load_dataset()
    msgs, labels = zip(*dataset)

    vectorizer = CountVectorizer()
    X = vectorizer.fit_transform(msgs)
    y = labels

    model = MultinomialNB()
    model.fit(X, y)

    joblib.dump(model, MODEL_FILE)
    joblib.dump(vectorizer, VEC_FILE)

    print("\n✅ Model Trained Successfully! Saved as spam_model.pkl")
    print("✅ Vectorizer Saved as vectorizer.pkl\n")
```

```
# ---------------------------
# Predict
# ---------------------------
def predict_spam(message):
    if not os.path.exists(MODEL_FILE) or not os.path.exists(VEC_FILE):
        print("❌ No trained model found! Please run: python spamdetector.py --train")
        return

    model = joblib.load(MODEL_FILE)
    vectorizer = joblib.load(VEC_FILE)

    X = vectorizer.transform([message])
    prediction = model.predict(X)[0]

    if prediction == "spam":
        print("🚨 SPAM DETECTED!")
    else:
        print("✔️ Message is Safe (HAM)")

# ---------------------------
# Interactive Mode
# ---------------------------
def interactive_mode():
    print("\n🎯 Interactive Mode Activated")
    print("Type a message to check if it's spam (type 'exit' to quit)\n")

    if not os.path.exists(MODEL_FILE):
        print("❌ No trained model found! Please run: python spamdetector.py --train")
        return

    while True:
        msg = input("Enter message: ")
        if msg.lower() == "exit":
            print("👋 Exiting interactive mode...")
            break
        predict_spam(msg)

# ---------------------------
# Argument Parser
# ---------------------------
parser = argparse.ArgumentParser()
parser.add_argument("--train", action="store_true", help="Train the spam model")
parser.add_argument("--interactive", action="store_true", help="Start interactive mode")
args = parser.parse_args()

if args.train:
    train_model()
elif args.interactive:
    interactive_mode()
else:
    print("\nUsage:")
    print(" python spamdetector.py --train        → Train the model")
    print(" python spamdetector.py --interactive  → Start chat mode\n")
]
```

House Price Prediction:

[

```
# CodexIntern AI/ML Task 3


import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.datasets import fetch_california_housing

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from sklearn.linear_model import LinearRegression

from sklearn.metrics import mean_squared_error, r2_score


# Load dataset

housing = fetch_california_housing()

data = pd.DataFrame(housing.data, columns=housing.feature_names)

data["PRICE"] = housing.target


print("Dataset Shape:", data.shape)

print("\nSample Rows:\n", data.head())


# Basic statistics

print("\nStats:\n", data.describe())


# Price distribution

plt.figure(figsize=(8,5))

plt.hist(data["PRICE"], bins=30, edgecolor='black')

plt.title("Price Distribution")
plt.xlabel("Price")
plt.ylabel("Count")
plt.show()

# Correlation heatmap
plt.figure(figsize=(12,8))
plt.imshow(data.corr(), cmap='coolwarm')
plt.colorbar()
plt.title("Correlation Heatmap")
plt.show()

# Split features and target
X = data.drop("PRICE", axis=1)
y = data["PRICE"]

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

# Scaling
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Model
model = LinearRegression()
model.fit(X_train_scaled, y_train)
```

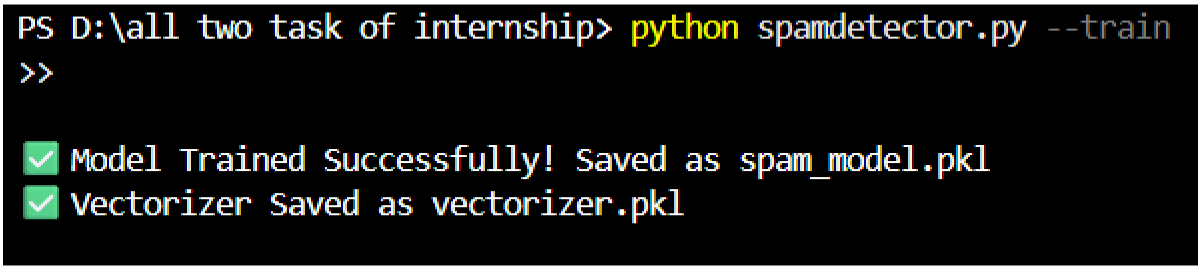
```
# Predictions
y_pred = model.predict(X_test_scaled)

# Metrics
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("\nModel Performance:")
print("MSE:", round(mse, 3))
print("R2 Score:", round(r2, 3))

# Visualization
plt.figure(figsize=(8,6))
plt.scatter(y_test, y_pred)
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red')
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title("Actual vs Predicted Price")
plt.show()

# Example prediction
sample = X.iloc[0]
sample_scaled = scaler.transform([sample])
prediction = model.predict(sample_scaled)[0]

print("\nPredicted Price for Sample House:", round(prediction, 3))

]
```

6. Results / Analysis

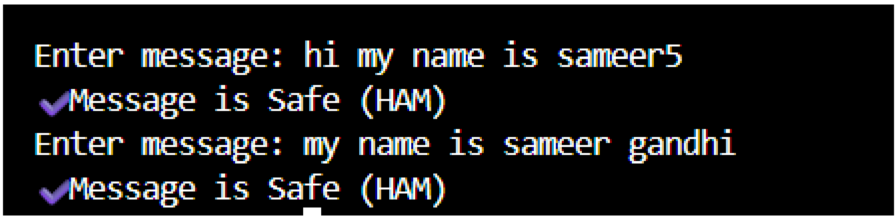
Spam Mail Detector:

- Accuracy, Precision, F1-score achieved
- Sample predictions for messages

[



```
PS D:\all two task of internship> python spamdetector.py --train
>>

✅ Model Trained Successfully! Saved as spam_model.pkl
✅ Vectorizer Saved as vectorizer.pkl
```



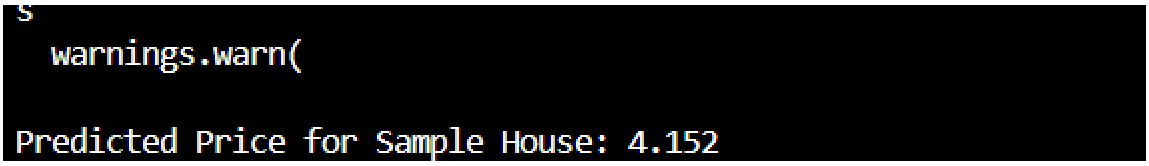
```
 Enter message: hi my name is sameer5
 ✔Message is Safe (HAM)
 Enter message: my name is sameer gandhi
 ✔Message is Safe (HAM)
```

]

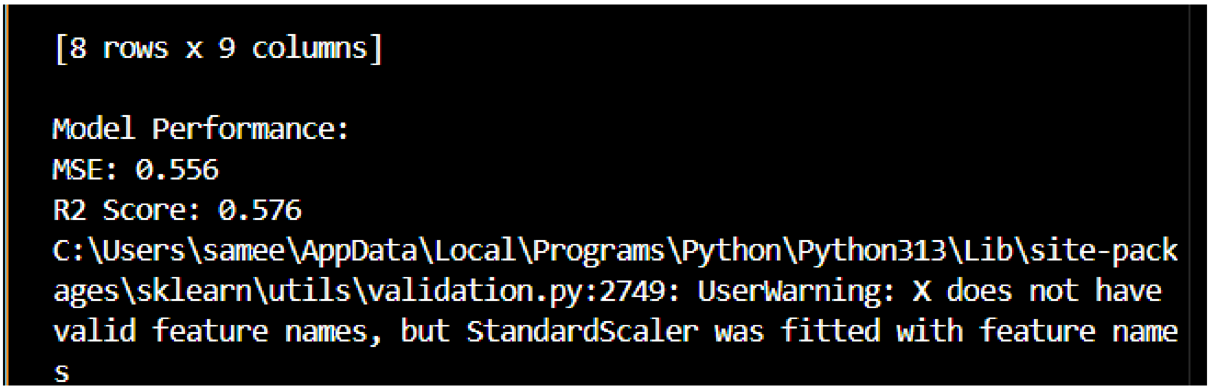
House Price Prediction:

- Model performance: MSE and R^2 score
- Example predicted house prices

[



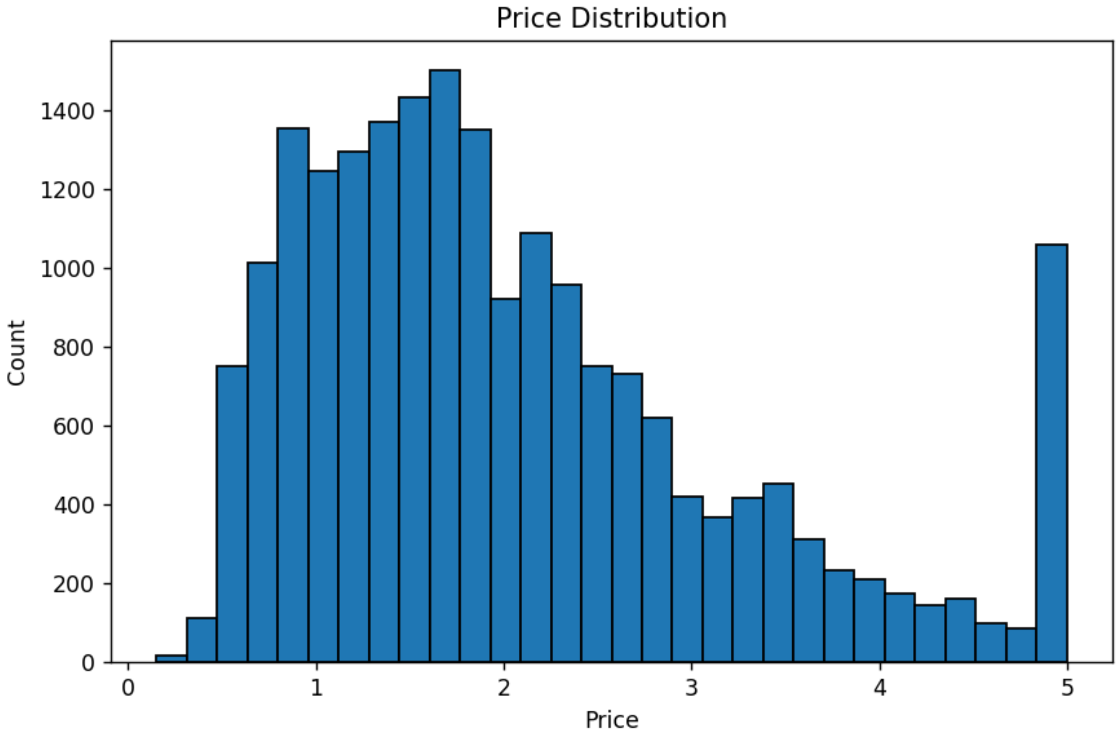
```
s
  warnings.warn(

Predicted Price for Sample House: 4.152
```

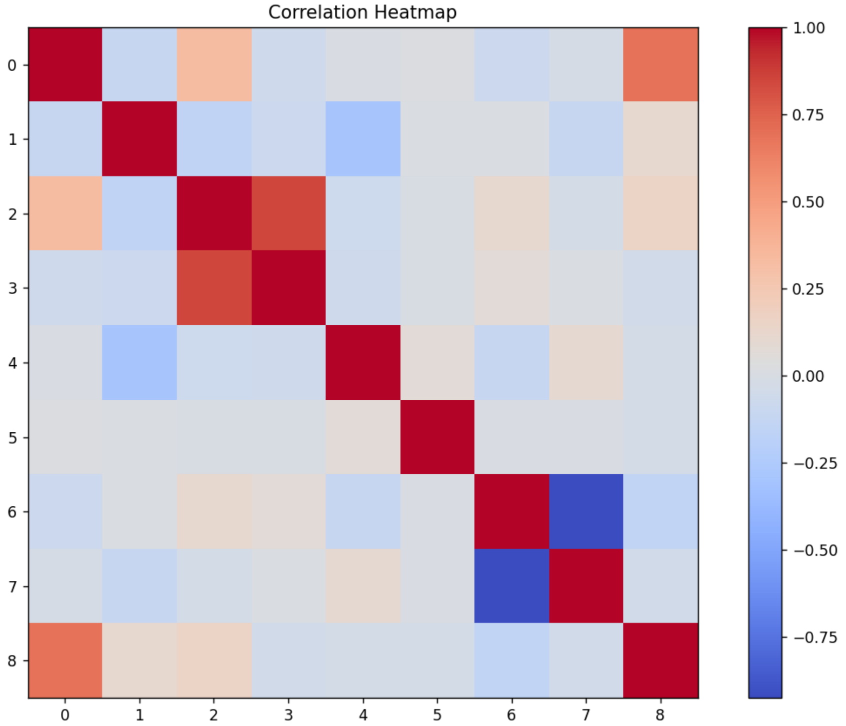


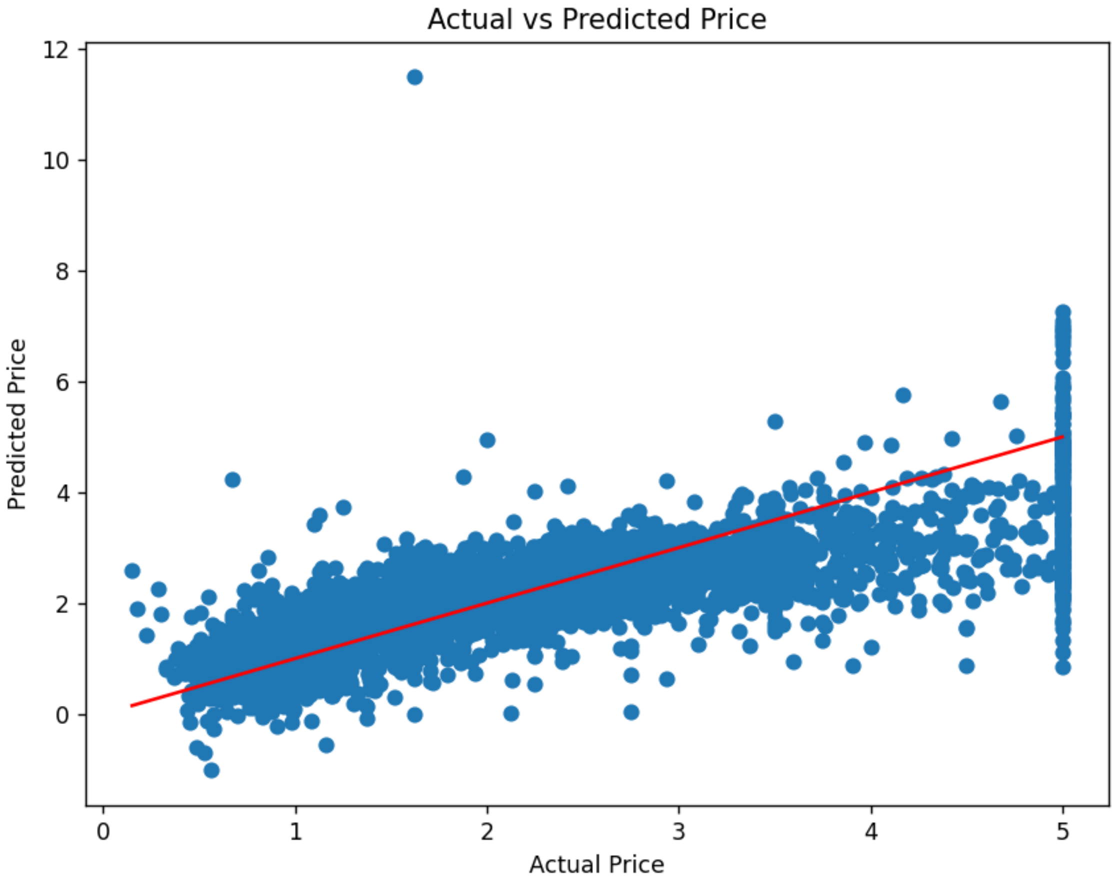
```
  [8 rows x 9 columns]

  Model Performance:
  MSE: 0.556
  R2 Score: 0.576
  C:\Users\samee\AppData\Local\Programs\Python\Python313\Lib\site-pack
  ages\sklearn\utils\validation.py:2749: UserWarning: X does not have
  valid feature names, but StandardScaler was fitted with feature name
  s
```

]

[


Price Distribution
Count
Price
0
200
400
600
800
1000
1200
1400
0
1
2
3
4
5


Correlation Heatmap
0
1
2
3
4
5
6
7
8
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75



]

7. Conclusion

Both tasks successfully demonstrate the application of AI & ML in real-world problems. Text preprocessing, feature extraction, classification, and regression modeling were applied effectively. The projects provide a solid foundation for advanced ML tasks.

8. Skills Gained

- Python programming for AI & ML
- Data preprocessing and visualization
- Text classification and regression modeling
- Model evaluation and interpretation of results

9. References

1. UCI SMS Spam Collection: https://archive.ics.uci.edu/ml/datasets/sms+spam+collection
2. California Housing Dataset: https://www.kaggle.com/camnugent/california-housing-prices
3. Scikit-learn Documentation: https://scikit-learn.org
4. NLTK Documentation: https://www.nltk.org