Assignment No -1

```
import pandas as pd


# Load the dataset (replace 'filename.csv' with your actual file name or path)
df = pd.read_csv('When2Heat_Heating_Profiles.csv')


# 1. Shape of the dataset
print("Shape of the dataset:", df.shape)


# 2. Check for missing values
print("\nMissing values per column:")
print(df.isnull().sum())


# 3. Data types of each column
print("\nData types of each column:")
print(df.dtypes)
```

Assignment No -2

```
# Import necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Enable inline plotting
%matplotlib inline

# Step 1: Create or load the dataset (Prices in rupees)
data = {
 'SquareFeet': [500, 750, 1000, 1250, 1500, 1750, 2000, 2250, 2500, 2750, 3000],
 'Bedrooms': [1, 2, 2, 3, 3, 4, 4, 4, 5, 5, 6],
 'Bathrooms': [1, 1, 2, 2, 2, 3, 3, 3, 4, 4, 5],
 'Price': [8000000, 12000000, 16000000, 20000000, 24000000, 28000000, 32000000, 36000000, 40000000, 44000000, 48000000]
}

# Convert the dataset into a DataFrame
df = pd.DataFrame(data)

# Step 2: Prepare the data
# Features: Square footage, number of bedrooms, and number of bathrooms
# Target: Price
X = df[['SquareFeet', 'Bedrooms', 'Bathrooms']].values
y = df['Price'].values

# Add a column of ones to X for the intercept term
```

```
X_b = np.c_[np.ones((X.shape[0], 1)), X]  # Shape: (n_samples, n_features+1)

# Step 3: Implement Linear Regression manually using the Normal Equation
# θ = (X^T * X)^(-1) * X^T * y
theta_best = np.linalg.inv(X_b.T @ X_b) @ X_b.T @ y

# Step 4: Extract the model parameters
intercept = theta_best[0]
coefficients = theta_best[1:]
print("Intercept (b):", intercept)
print("Coefficients (m):", coefficients)

# Step 5: Make predictions
def predict(X, theta):
   return X @ theta

y_pred = predict(X_b, theta_best)

# Step 6: Evaluate the model
# Mean Squared Error (MSE)
mse = np.mean((y - y_pred) ** 2)
print(f"Mean Squared Error (MSE): ₹{mse:.2f}")

# Step 7: Visualize the Predictions vs Actual Prices
plt.figure(figsize=(8, 6))
plt.scatter(range(len(y)), y, color='blue', label='Actual Prices')
plt.scatter(range(len(y_pred)), y_pred, color='red', label='Predicted Prices',
alpha=0.7)
```

```
plt.title('Actual vs Predicted House Prices (in ₹)', fontsize=16)
plt.xlabel('House Index', fontsize=14)
plt.ylabel('Price (₹)', fontsize=14)
plt.legend(fontsize=12)
plt.grid(True)
plt.show()
```

Assignment No -3

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt


import os

from skimage.io import imread

from skimage.transform import resize


plt.figure(figsize=(10,5))

img1 = r"C:\Users\Ishwari\Downloads\test_set\test_set\dogs\dog.4450.jpg"

plt.imshow(imread(img1))


img_path = r"C:\Users\Ishwari\Downloads\test_set\test_set\dogs\dog.5000.jpg"

img = imread(img_path)

img


img_resize = resize(img, (15,15))

img_resize.shape


flatten_img = img_resize.flatten()

flatten_img


input_dir = r"C:\Users\Ishwari\Downloads\test_set\test_set"

categories = ['cats', 'dogs']

data = []

labels = []
```

```
for category_idx, category in enumerate(categories):
    for file in os.listdir(os.path.join(input_dir, category)):
        img_path = os.path.join(input_dir, category, file)
        print(img_path)
        img = imread(img_path)
        img = resize(img, (15,15))
        data.append(img.flatten())
        labels.append(category_idx)

data = np.asarray(data)
labels = np.asarray(labels)

from sklearn.model_selection import train_test_split

from sklearn.svm import SVC
svm_model = SVC()

svm_model.fit(x_train, y_train)

y_pred = svm_model.predict(x_test)
y_pred

from sklearn.metrics import accuracy_score
score = accuracy_score(y_test, y_pred)
score

from sklearn.model_selection import cross_val_score
cross_val_score = cross_val_score(svm_model, data, labels, cv = 5)
```

```
cross_val_score


Mean_Accuracy = cross_val_score.mean()
Mean_Accuracy


from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))


from sklearn.model_selection import GridSearchCV


classifer = SVC()
parameters = [{'gamma':[0.01, 0.001, 0.0001], 'C':[10, 100, 1000]}]
grid_search = GridSearchCV(classifer, parameters)
grid_search.fit(x_train, y_train)


best_estimator = grid_search.best_estimator_
best_estimator


y_prediction = best_estimator.predict(x_test)
y_prediction


plt.figure(figsize=(10, 5))
img_path = r"C:\Users\Ishwari\Downloads\test_set\test_set\dogs\dog.5000.jpg"
plt.imshow(imread(img1))


import cv2 as cv
img_path = cv.imread(img1)
plt.imshow(img_path)
```

```
img_dog = r"C:\Users\Ishwari\Downloads\test_set\test_set\dogs\dog.5000.jpg"
img_path = cv.imread(img_dog)
plt.imshow(img_path)

img_dog = r"C:\Users\Ishwari\Downloads\test_set\test_set\dogs\dog.5000.jpg"
img_new = imread(img_dog)
img_new1 = resize(img_new, (15,15))
img_flatten = img_new1.flatten()
img_array = np.asarray(img_flatten)

result = svm_model.predict(img_array.reshape(1, -1))

if result[0] == 1:
   print("Result =", result[0])
   print("It is a cat.")
else:
   print("It is a dog.")

img2 = r"C:\Users\Ishwari\Downloads\test_set\test_set\dogs\dog.4600.jpg"
image_classification_prediction(img2)

img3 = r"C:\Users\Ishwari\Downloads\test_set\test_set\cats\cat.4700.jpg"
image_classification_prediction(img3)
```

Assignment No -4

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import Pipeline
from sklearn.metrics import classification_report, accuracy_score

# Load dataset
df = pd.read_csv("spam.csv", encoding='latin-1')[['v1', 'v2']]
df.columns = ['label', 'message']

# Convert labels to binary values
df['label'] = df['label'].map({'ham': 0, 'spam': 1})

# Split dataset
X_train, X_test, y_train, y_test = train_test_split(df['message'], df['label'],
test_size=0.2, random_state=42)

# Create pipeline: TF-IDF + Naive Bayes
model = Pipeline([
  ('tfidf', TfidfVectorizer(stop_words='english')),
  ('clf', MultinomialNB())
])

# Train the model
```

```
model.fit(X_train, y_train)


# Evaluate



y_pred = model.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))

print("Report:\n", classification_report(y_test, y_pred))


# Test the model

def predict_sms(text):

  result = model.predict([text])

  return "Spam" if result[0] == 1 else "Ham"


# Example

print(predict_sms("Congratulations! You've won a $1,000 Walmart gift card. Call now!"))
```