Task 2: Data Scaling and Dividing

"""

Title: Final Project

Author: Stephen Marth

Date: 15 March 2025

Version: Task 2: Data Scaling and Dividing

"""

https://www.kaggle.com/datasets/shantanugarg274/lung-cancer-prediction-dataset << Data Set

```
import numpy as np

import os

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

import urllib.request

import os

import seaborn as sns


#github

url = "https://raw.githubusercontent.com/Stephengmarth/ISE_221/main/Lung%20Cancer%20Dataset.csv"

local_file = "lung_cancer_dataset.csv"

file_name = os.path.basename(local_file)

feature_threshold = 0.15  #threshold to dertimine which features to use
```

```
#function to help determine which features to use Pearson Correlation Formula
def feature_correlation(X, y, feature_name):
    correlations = []
    for i in range(X.shape[1]):  # Loop through features
        cor = np.corrcoef(X[:, i], y)[0, 1]  #(Feature[],target)[]
        correlations.append((feature_name[i], cor))  #store feature name and correlation
    correlations.sort(key=lambda x: abs(x[1]), reverse=True)#sort

    print(f"\nFeature Correlation with {target}:")
    for feature, cor in correlations:
        print(f"{feature}: {cor:.4f}")

    return correlations

#csv uploaded to github was going to use kaggle but needed my creds
url =
"https://raw.githubusercontent.com/Stephengmarth/ISE_221/main/Lung%20Cancer%20Dataset.csv"
local_file = "lung_cancer_dataset.csv"

#download from github
try:
    if not os.path.exists(local_file):
        urllib.request.urlretrieve(url, local_file)
except Exception as e: #e throws the exception
    print("Cant Download CSV", e)
    exit()
```

```
data = np.genfromtxt(local_file, delimiter=',', skip_header=1, dtype=str, encoding="utf-8")


# change words to number, still a string

data[data == "YES"] = "1"

data[data == "NO"] = "0"

#change data type

data = data.astype(float)

y = data[:, -1]

X = data[:, :-1]


#print to make sure it was done right

#print("this is after conversion of yes and no to 1 and 0",data[:5])


with open(local_file, "r") as file:

   header = file.readline().strip().split(",")



target = header[-1]


#print(data.shape)

print (f"{file_name} has\n{data.shape[1]} Features\n{data.shape[0]} Samples") #use f"
embed variables in statement

#get a features list take hader seperate by ,


#print feature list loop with index

print("Feature List with Index:")
```

```
index = 0
for feature in header:
   print(f"[{index}] {feature}")
   index += 1


#check for missing data with numpy.isnan(name).sum()
missing_values = np.isnan(data).sum()
print(f"There are {missing_values} missing values.") # 5000 are the string yes or no



unique, counts = np.unique(y, return_counts=True)#finds values of 1 and 0 and total
plt.figure(figsize=(6, 4))
plt.bar(unique, counts, color=['blue', 'red'])
plt.xticks([0, 1], labels=["Negative", "Positive"])
plt.xlabel(target)
plt.ylabel("Number of Samples")
plt.title(f"Target Variable in {file_name}")
plt.tight_layout()
plt.savefig("target_distribution.png")
plt.show()


correlations = feature_correlation(X, y, header[:-1])



#Choosing features to use.
#based on r value interpritiation I dont have anything with a srong linear corliation. so we are going to set the bar low. might be best to use all.
```

```
selected_features = [feature for feature, corr in correlations if abs(corr) > feature_threshold]


# Print selected features

print(f"\n These Features are above a correlation threshold of {feature_threshold}:")

print(selected_features)


#okay i got six whats next


predictive_feature_indices = [header.index(feature) for feature, corr in correlations if abs(corr) > feature_threshold]

X_selected = X[:, predictive_feature_indices]


print(f"Predictive Features update shape: {X_selected.shape}")

#sick


 #end with visual bar chart of selected figures


predictive_features = [feature for feature, corr in correlations if abs(corr) > feature_threshold]

r_values = [corr for feature, corr in correlations if abs(corr) > feature_threshold]


# Plot bar chart for selected features

plt.figure(figsize=(12, 6))

plt.bar(predictive_features, r_values, color='blue', alpha=1)

plt.xlabel("Features")

plt.ylabel("Correlation")
```

```
plt.title(f"Feature Correlation with {target} above {feature_threshold} threshold")
plt.xticks(rotation=45)
plt.tight_layout()
plt.savefig("correlation_bar_chart.png")
plt.show()

#standardize
means = np.mean(X_selected, axis=0)
stds = np.std(X_selected, axis=0)

#in case division by zero
stds[stds == 0] = 1

X_scaled = (X_selected - means) / stds

print("should be near 0):", np.mean(X_scaled, axis=0))
print("Standard diviation:", np.std(X_scaled, axis=0))

#60/40
X_train, X_temp, y_train, y_temp = train_test_split(
   X_scaled, y, test_size=0.4, random_state=29)

#20/20
X_val, X_test, y_val, y_test = train_test_split(
   X_temp, y_temp, test_size=0.5, random_state=36)
```

```
#lets see what we got
print("Training Data Set:", X_train.shape, y_train.shape)
print("Validation Data Set:", X_val.shape, y_val.shape)
print("Test Data Set:", X_test.shape, y_test.shape)

sizes = [len(y_train), len(y_val), len(y_test)]
labels = ["Training", "Validation", "Testing"]
colors = ['blue', 'lightblue', 'lightgreen']

plt.figure(figsize=(6, 6))
plt.pie(sizes, labels=labels, autopct='%1.1f%%', colors=colors, startangle=140)
plt.title("Dataset Split")
plt.axis('equal')
plt.savefig("dataset_split_pie.png")
plt.show()

#imma save this
train_data = np.column_stack((X_train, y_train))
val_data = np.column_stack((X_val, y_val))
test_data = np.column_stack((X_test, y_test))

selected_headers = [header[i] for i in predictive_feature_indices]
all_headers = selected_headers + [target]

np.savetxt("train_data.csv", train_data, delimiter=",", header=",".join(all_headers),
comments='', fmt='%.4f')
```

```
np.savetxt("validation_data.csv", val_data, delimiter=",", header=",".join(all_headers), comments='', fmt='%.4f')

np.savetxt("test_data.csv", test_data, delimiter=",", header=",".join(all_headers), comments='', fmt='%.4f')


plt.figure(figsize=(12, 8))


for i in range(X_scaled.shape[1]):
    plt.subplot(2, 3, i + 1)
    sns.histplot(X_scaled[:, i], kde=True, color='skyblue', bins=30)
    plt.title(predictive_features[i])
    plt.xlabel("Scaled Value")
    plt.ylabel("Density")

plt.suptitle("Feature Distributions After Standardization", fontsize=16)
plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.savefig("standardization_values")
plt.show()
```