Assignment - 4

Machine Learning (CS 5710) CRN: 23922

Name: Midhun Kumar Chintapalli

700739502

Github link: <https://github.com/midhun-ch/ML_Assignment4>

Video link : <https://drive.google.com/file/d/1cGgVutsoUtM8E_HbC8t2kBlSnRcjHUqq/view?usp=share_link>

**1.Pandas**

from matplotlib import pyplot as plt

import pandas as pd

import numpy as np

**# Read the provided CSV file ‘data.csv’**

data = pd.read\_csv('data.csv')

**# Show the basic statistical description about the data**

print(data.describe(), '\n')

**# Check if the data has null values.**

print("Null values in the data: \n", data.isnull().sum(), '\n')

**# a. Replace the null values with the mean**

data.fillna(data.mean(), inplace=True)

print("Null values in the data after replacing with mean: \n", data.isnull().sum(), '\n')

print(data, '\n')

**# Select at least two columns and aggregate the data using: min, max, count, mean**

**# selecting two columns pulse and calories**

print("Aggregating the data using min, max, count, mean: \n", data[['Pulse', 'Calories']].agg(['min', 'max', 'count', 'mean']), '\n')

**# Filter the dataframe to select the rows with calories values between 500 and 1000**

print("Filtering the dataframe to select the rows with calories values between 500 and 1000: \n", data[(data['Calories'] > 500) & (data['Calories'] < 1000)], '\n')

**# Filter the dataframe to select the rows with calories values > 500 and pulse < 100**

print("Filtering the dataframe to select the rows with calories values > 500 and pulse < 100: \n", data[(data['Calories'] > 500) & (data['Pulse'] < 100)], '\n')

**# Create a new “df\_modified” dataframe that contains all the columns from df except for “Maxpulse”**

df\_modified = data.drop('Maxpulse', axis=1)

print("New dataframe after dropping Maxpulse column: \n", df\_modified, '\n')

**# Delete the “Maxpulse” column from the main df dataframe**

data.drop('Maxpulse', axis=1, inplace=True)

print("Dataframe after dropping Maxpulse column: \n", data, '\n')

**# Convert the datatype of Calories column to int datatype**

data['Calories'] = data['Calories'].astype(int)

print("Data types of all columns after converting Calories to int: \n", data.dtypes, '\n')

**# Using pandas create a scatter plot for the two columns (Duration and Calories)**

data.plot.scatter(x='Duration', y='Calories', title='Scatter plot for Duration and Calories')

plt.show()

**Titanic Dataset**

from matplotlib import pyplot as plt

import pandas as pd

import numpy as np

from sklearn.naive\_bayes import GaussianNB

from sklearn.metrics import accuracy\_score

train\_data = pd.read\_csv('train\_preprocessed.csv')

test\_data = pd.read\_csv('test\_preprocessed.csv')

**# Find the correlation between Survived and Sex**

corr = train\_data['Survived'].corr(train\_data['Sex'].astype('category').cat.codes)

print("Correlation between Survived and Sex: ",corr)

print('a. Do you think we should keep this feature?')

print('Yes, we should keep this feature as it has a correlation of', corr, 'with the target variable but some other features can be dropped as they have very less correlation with the target variable.')

**# Do at least two visualizations to describe the data**

**# Histogram of age**

train\_data['Age'].plot.hist(title='Histogram of Age')

plt.show()

**# Scatter plot of age and fare**

train\_data.plot.scatter(x='Age', y='Fare', title='Scatter plot of Age and Fare')

plt.show()

**# Plot between age and survived**

train\_data.plot.scatter(x='Age', y='Survived', title='Scatter plot of Age and Survived')

plt.show()

**# Implement Naïve Bayes method using scikit-learn library and report the accuracy**

**# Create a Gaussian Classifier**

model = GaussianNB()

**# Train the model using the training sets which is already preprocessed**

**# drop PassengerId column as it is not required in test data**

test\_data.drop('PassengerId', axis=1, inplace=True)

**# drop Survived column in train data as it is the target variable**

train\_data.drop('Survived', axis=1, inplace=True)

**# Fit the model with the training data**

model.fit(train\_data.drop('Embarked', axis=1), train\_data['Embarked'])

**# Predict the response for test dataset**

y\_pred = model.predict(test\_data.drop('Embarked', axis=1))

**# Calculate the accuracy of the model**

print("Accuracy of the model: ", accuracy\_score(test\_data['Embarked'], y\_pred))

**Glass Dataset**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.naive\_bayes import GaussianNB

from sklearn.svm import SVC

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import classification\_report, accuracy\_score

import warnings

warnings.filterwarnings("ignore")

**# a. read glass.csv file as a dataframe**

glass\_data = pd.read\_csv('glass.csv')

**# b. Use train\_test\_split to create training and testing part**

**# Split the data into training and testing data**

X\_train, X\_test, y\_train, y\_test = train\_test\_split(glass\_data.drop('Type', axis=1), glass\_data['Type'], test\_size=0.3, random\_state=42)

**# implement Naïve Bayes method using scikit-learn library and Evaluate the model on testing part using score and classification\_report**

**# Create a Gaussian Classifier**

model = GaussianNB()

**# Train the model using the training sets**

model.fit(X\_train, y\_train)

**# Predict the response for test dataset**

y\_pred = model.predict(X\_test)

**# Calculate the accuracy of the model**

print("Accuracy of the Naive Bayes model: ", accuracy\_score(y\_test, y\_pred))

**# Print classification report**

print('Classification Report for Naive Bayes model: ')

print(classification\_report(y\_test, y\_pred))

**# Use SVM method using scikit-learn library and Evaluate the model on testing part using score and classification\_report**

**# Create a SVM Classifier**

model = SVC(kernel='linear')

**# Train the model using the training sets**

model.fit(X\_train, y\_train)

**# Predict the response for test dataset**

y\_pred = model.predict(X\_test)

**# Calculate the accuracy of the model**

print("Accuracy of the SVM model: ", accuracy\_score(y\_test, y\_pred))

**# Print classification report**

print('Classification Report for SVM model: ')

print(classification\_report(y\_test, y\_pred))

**# Do at least two visualizations to describe or show correlations in the Glass Dataset**

**# Histogram of refractive index**

glass\_data['RI'].plot.hist(title='Histogram of Refractive Index')

plt.show()

**# Scatter plot of refractive index and Ca**

glass\_data.plot.scatter(x='RI', y='Ca', title='Scatter plot of Refractive Index and Ca')

plt.show()

print('Which algorithm you got better accuracy? Can you justify why?')

**#accuracy of Naive Bayes model: 0.3076923076923077**

**#accuracy of SVM model: 0.676923076923077**

print('SVM model has better accuracy than Naive Bayes model. This is because SVM model tries to find the best possible decision boundary between the data points of different classes. It tries to maximize the margin between the decision boundary and the data points. On the other hand, Naive Bayes model assumes that the features are independent of each other and tries to find the probability of the data point belonging to a particular class. Hence, SVM model has better accuracy than Naive Bayes model.')