JSS MAHAVIDYAPEETHA

Lab file

Subject Name: Data Analysis lab

Lab Subject Code: KIT 651

COURSE : B.Tech
SEMESTER : VI SEM

Name	**Kushagra Arora**
Roll No.	**1900910130065**
Section-Batch	**IT1 batch 21-22**

Department of Engineering
JSS ACADEMY OF TECHNICAL EDUCATION

C-20/1, SECTOR-62, NOIDA

VISION OF THE INSTITUTE

JSS Academy of Technical Education Noida aims to become an Institution of excellence in imparting quality Outcome Based Education that empowers the young generation with Knowledge, Skills,Research, Aptitude and Ethical values to solve Contemporary Challenging Problems.

MISSION OF THE INSTITUTE

- Develop a platform for achieving a globally acceptable level of intellectual acumen and technological competence.
- Create an inspiring ambience that raises the motivation level for conducting quality research.
- Provide an environment for acquiring ethical values and a positive attitude.

VISION OF THE DEPARTMENT

To spark the imagination of the Information technology Engineers with values, skills and creativity to solve real world problems.

MISSION OF THE DEPARTMENT

- To inculcate creative thinking and problem solving skills through effective teaching, learning and research.
- To empower professionals with core competency in the field of INFORMATION TECHNOLOGY.
- To foster independent and lifelong learning with ethical and social responsibilities.

INDEX

1. WAP to implement numerical operations in python

Theory

operation	result	example
max()	Finds max element in arr	max(2,3,51,32) returns 51
min()	Finds min element in arr	max(2,3,51,32) returns 2
sum()	Finds sum of all elements	sum(2,3,51,32) returns 88
avg()	Finds avg of all elements	avg(2,3,51,32) returns 17.6
sqrt()	Finds sqrt of all elements	sqrt(51) returns 7.14
round()	Finds round of number	round(34.232,1) returns 34.2

Code

```
from audioop import avg
from cmath import sqrt
import math
arr = list(map(int,input().split()))
print(arr)
print("maximum of numbers " ,max(arr))
print("minimum of numbers " ,min(arr))
print("sum of numbers " ,sum(arr))
avg = sum(arr)/len(arr)
print("average of numbers " ,avg)
n = int(input("enter number to find sqrt of "))
print(sqrt(n))
p = float(input("enter number to find round off of "))
print(round(n,2))
```

2. WAP to implement statistical operations in python

Theory

Import necessary libraries

operation	result
median	Finds median element in arr
mean	Finds mean element in arr
mode	Finds mode of all elements
stdev	Find standard deviation in arr

Code

```
import math
import statistics
arr = list(map(int,input().split()))
print(arr)
print ("The median values is : ",end="")
print (statistics.median(arr))
print ("The mean values is : ",end="")
print (statistics.mean(arr))
print ("The mode values is : ",end="")
print (statistics.mode(arr))
print ("The Standard deviation is : ")
print (statistics.stdev(arr))
```

3. WAP to implement data handling operations in python

Theory

Pandas DataFrames make manipulating your data easy, from selecting or replacing columns and indices to reshaping data.

Two-dimensional, size-mutable, potentially heterogeneous tabular data.
Data structure also contains labeled axes (rows and columns). Arithmetic operations align on both row and column labels

Functions

Read_csv

Read a comma-separated values (csv) file into DataFrame.

Index: Index or array-like

Index to use for resulting frame. Will default to RangeIndex if no indexing information part of input data and no index provided.

Columns: Index or array-like

Column labels to use for resulting frame when data does not have them, defaulting to RangeIndex(0, 1, 2, …, n). If data contains column labels, will perform column selection instead.

Dtype: dtype, default None

Data type to force. Only a single dtype is allowed. If None, infer.

Copy: bool or None, default None

Copy data from inputs. For dict data, the default of None

Loc

Access a group of rows and columns by label(s) or a boolean array.

Shape

Return a tuple representing the dimensionality of the DataFrame.

size

Return an int representing the number of elements in this object.

to_csv([path_or_buf, sep, na_rep, ...])

Write object to a comma-separated values (csv) file.

Code - deleting missing rows

```
import pandas as pd
import pandas as pd
df =
pd.read_csv("https://raw.githubusercontent.com/kastrahl/Data-science/main/adult
.csv")
df.head()
df.columns = [ 'age', 'workclass', 'fnlwgt', 'education', 'education-num',
'marital-status', 'occupation', 'relationship', 'race', 'sex', 'capital-gain',
'capital-loss', 'hours-per-week', 'native-country', 'salary']
df.head
df.size
df.shape # get the array dimensions
df['education'] # get column data based on column name
df.loc[0]  #get row data based on row index
df.loc[1:5, 'age':'occupation'] #get subset of dataframe [row, column]
df.loc[df['age'] == 45] # extract rows where col1 has age = 45
# get descriptive statistics (count, mean, std dev, min, max, and quartiles)
# it excludes all NaN or missing values
df.describe()
df1 = df.loc[1:5, 'age':'occupation'] #get subset of dataframe [row, column]
df1.style # view saved dataframe
df1.to_csv("df1.csv")
exit
```

4. WAP to implement data pre-processing operations in python

Theory

Steps Involved in Data Preprocessing:

1. Data Cleaning:

The data can have many irrelevant and missing parts. To handle this part, data cleaning is done. It involves handling of missing data, noisy data etc.

(a). Missing Data:

This situation arises when some data is missing in the data. It can be handled in various ways. Some of them are:

1. **Ignore the tuples:**

   This approach is suitable only when the dataset we have is quite large and multiple values are missing within a tuple.

2. **Fill the Missing values:**

   There are various ways to do this task. You can choose to fill the missing values manually, by attribute mean or the most probable value.

(b). Noisy Data:

Noisy data is a meaningless data that can't be interpreted by machines.It can be generated due to faulty data collection, data entry errors etc. It can be handled in following ways :

1. **Regression:**

   Here data can be made smooth by fitting it to a regression function.The regression used may be linear (having one independent variable) or multiple (having multiple independent variables).

2. **Clustering:**

   This approach groups the similar data in a cluster. The outliers may be undetected or it will fall outside the clusters.

2. Data Transformation:

This step is taken in order to transform the data in appropriate forms suitable for mining process. This involves following ways:

1. **Normalization:**

It is done in order to scale the data values in a specified range (-1.0 to 1.0 or 0.0 to 1.0)

2. **Attribute Selection:**

In this strategy, new attributes are constructed from the given set of attributes to help the mining process.

3. **Data Reduction:**

Since data mining is a technique that is used to handle huge amount of data. While working with huge volume of data, analysis became harder in such cases. In order to get rid of this, we uses data reduction technique. It aims to increase the storage

efficiency and reduce data storage and analysis costs. The various steps to data reduction are:

1. **Attribute Subset Selection:**
   The highly relevant attributes should be used, rest all can be discarded.
2. **Dimensionality Reduction:**
   This reduce the size of data by encoding mechanisms.It can be lossy or lossless. If after reconstruction from compressed data, original data can be retrieved, such reduction are called lossless reduction else it is called lossy reduction. The two effective methods of dimensionality reduction are:Wavelet transforms and PCA (Principal Component Analysis).

CODE -

```
"""DA4.ipynb
Automatically generated by Colaboratory.
Original file is located at
https://colab.research.google.com/drive/16myUt6q-IwahQk8OT0xyyAKUvUKNPlnU
"""
import pandas as pd
df =
pd.read_csv("https://raw.githubusercontent.com/kastrahl/Data-science/main/train.csv")
df.head()
import numpy as np
df.drop("Name",axis=1,inplace=True)
df.drop("Ticket",axis=1,inplace=True)
df.drop("PassengerId",axis=1,inplace=True)
df.drop("Cabin",axis=1,inplace=True)
df.drop("Embarked",axis=1,inplace=True)
df.head()
df.info()
#age is null for a lot of rows
print(df.isnull().sum())
#sklearn label encoder
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['Sex'] = le.fit_transform(df['Sex'])
newdf=df
newdf.head
newdf.info()
""" Deleting the column with missing data"""
```

```
df1 = df.dropna(axis=1) #deletes any column with value null
df1.info()
y1 = df1['Survived']
df1.drop("Survived",axis=1,inplace=True)
y1.size
from sklearn import metrics
from sklearn.model_selection import train_test_split
X_train, X_test,y_train,y_test = train_test_split(df1,y1,test_size=0.3)
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(X_train,y_train)
pred = lr.predict(X_test)
print(metrics.accuracy_score(pred,y_test))
"""Deleting the row with missing data
axis=1 is used to drop the column with `NaN` values.
axis=0 is used to drop the row with `NaN` values.
"""
df2 = newdf.dropna(axis = 0)
df2.info()
y2 = df2['Survived']
df2.drop("Survived",axis=1,inplace=True)
y2.size
from sklearn import metrics
from sklearn.model_selection import train_test_split
X_train, X_test,y_train,y_test = train_test_split(df2,y2,test_size=0.3)
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(X_train,y_train)
pred = lr.predict(X_test)
print(metrics.accuracy_score(pred,y_test))
"""MIN - MAX NORMALISATION """
df3=newdf.dropna(axis = 0)
df3.info
normalized_df=(df3-df3.min())/(df3.max()-df3.min())
normalized_df.head()
"""LEMMATIZATION
USING WORDNET
"""
import nltk
nltk.download('wordnet')
from nltk.stem import WordNetLemmatizer
```

```
# Create WordNetLemmatizer object
wnl = WordNetLemmatizer()
# single word lemmatization examples
list1 = ['kites', 'babies', 'dogs', 'flying', 'smiling',
         'driving', 'died', 'tried', 'feet']
for words in list1:
    print(words + " ---> " + wnl.lemmatize(words))
"""stop word removal """
import io
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
# word_tokenize accepts a string as an input, not a file.
stop_words = set(stopwords.words('english'))
file1 = open("stop_word.txt")
#to read file content as a stream:
line = file1.read()
words = line.split()
for i in words:
    if not r in stop_words:
        appendFile = open('filteredtext.txt','a')
        appendFile.write(" "+r)
        appendFile.close()
```

5. WAP to PERFORM DIMENSIONALITY REDUCTION using PCA and test effect on SVM classifier with diabetes dataset

Theory

This reduce the size of data by encoding mechanisms.It can be lossy or lossless. If after reconstruction from compressed data, original data can be retrieved, such reduction are called lossless reduction else it is called lossy reduction. The two effective methods of dimensionality reduction are:Wavelet transforms and PCA (**Principal Component Analysis**).

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.
The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

Code-

```
# -*- coding: utf-8 -*-
"""da5.1.ipynb
Automatically generated by Colaboratory.
Original file is located at
    https://colab.research.google.com/drive/16LrZKkdkTaqWpn7dBA3EMpFbWVaNj2l-
"""
import numpy as np
import pandas as pd
from pandas_profiling import ProfileReport
from sklearn import svm
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
df =
pd.read_csv('https://raw.githubusercontent.com/kastrahl/Data-science/main/diabete
s.csv')
#display the head (first 5 rows) og the dataset
df.head()
# !pip install -U pandas-profiling
```

```
profile = ProfileReport(df)
profile
#! pip install
https://github.com/pandas-profiling/pandas-profiling/archive/master.zip
df.describe()
# Extract Features
X = df.iloc[:, :8]
X.head()
# Extract Class Labels
y = df["Outcome"]
y.head()
# Split Dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.75,
random_state=0)
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
# Normalize Features
scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
# View first 5 rows
X_train[:5, :]
"""testing which of the svm classifications , linear , polynomial , rbf or
sigmoidal gives us the best accuracy"""
# SVM Kernels
for k in ('linear', 'poly', 'rbf', 'sigmoid'):
    model = svm.SVC(kernel=k)
    model.fit(X_train, y_train)
    y_pred = model.predict(X_train)
    print(k)
    print(accuracy_score(y_train, y_pred))
# Using the best model
model = svm.SVC(kernel='rbf')
model.fit(X_train, y_train)
"""WITH PCA”””
from sklearn.decomposition import PCA # to apply PCA
```

```
import seaborn as sns # to plot the heat maps
df1 =
pd.read_csv('https://raw.githubusercontent.com/kastrahl/Data-science/main/diabete
s.csv')
#display the head (first 5 rows) og the dataset
df1.head()
# Extract Features from dataframe to
X = df1.iloc[:, :8]
X.head()
# Extract Class Labels
y = df1["Outcome"]
y.head()
# Split Dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.75,
random_state=0)
print(X_train.shape)
#Create an object of StandardScaler which is present in sklearn.preprocessing
scalar = StandardScaler()
scaled_data = pd.DataFrame(scalar.fit_transform(X_train)) #scaling the data
scaled_data
sns.heatmap(scaled_data.corr())
#Applying Principal Component Analysis we have taken n_components = 4, which
means our final feature set will have 3 columns
pca = PCA(n_components = 5)
pca.fit(scaled_data)
data_pca = pca.transform(scaled_data)
data_pca = pd.DataFrame(data_pca,columns=['PC1','PC2','PC3','PC4','PC5'])
data_pca.head()
#Checking Co-relation between features after PCA
sns.heatmap(data_pca.corr())
# SVM Kernels
for k in ('linear', 'poly', 'rbf', 'sigmoid'):
    model = svm.SVC(kernel=k)
    model.fit(data_pca, y_train)
    y_pred = model.predict(data_pca)
    print(k)
    print(accuracy_score(y_train, y_pred))
```

6. WAP to simple LINEAR REGRESSION on iris dataset

Theory

Regression models describe the relationship between variables by fitting a line to the observed data. Linear regression models use a straight line, while logistic and nonlinear regression models use a curved line. Regression allows you to estimate how a dependent variable changes as the independent variable(s) change.
Simple linear regression is used to estimate the relationship between two quantitative variables. You can use simple linear regression when you want to know:

- How strong the relationship is between two variables (e.g. the relationship between rainfall and soil erosion).
- The value of the dependent variable at a certain value of the independent variable (e.g. the amount of soil erosion at a certain level of rainfall).

Code-

```
# -*- coding: utf-8 -*-
"""DA6.ipynb
Automatically generated by Colaboratory.
Original file is located at
    https://colab.research.google.com/drive/1UmtcaL-RiPEKlRwqQxT5gl5MgTh-Arcn
"""
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
iris = load_iris()
iris_df = pd.DataFrame(data= iris.data, columns= iris.feature_names)
target_df = pd.DataFrame(data= iris.target, columns= ['species'])
def converter(specie):
    if specie == 0:
        return 'setosa'
    elif specie == 1:
        return 'versicolor'
    else:
        return 'virginica'
target_df['species'] = target_df['species'].apply(converter)
# Concatenate the DataFrames
```

```
iris_df = pd.concat([iris_df, target_df], axis= 1)
iris_df.describe()
iris_df.info()
import seaborn as sns
sns.pairplot(iris_df, hue= 'species')
import matplotlib.pyplot as plt
from sklearn import preprocessing, svm
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
# Converting Objects to Numerical dtype
iris_df.drop('species', axis= 1, inplace= True)
target_df = pd.DataFrame(columns= ['species'], data= iris.target)
iris_df = pd.concat([iris_df, target_df], axis= 1)
# Variables , indepent vs dependent variable
X= iris_df.drop(labels= 'sepal length (cm)', axis= 1)
y= iris_df['sepal length (cm)']
# Splitting the Dataset into training and test dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size= 0.33,
random_state= 101)
# Instantiating LinearRegression() Model
lr = LinearRegression()
# Training/Fitting the Model
lr.fit(X_train, y_train)
# Making Predictions
lr.predict(X_test)
pred = lr.predict(X_test)
# Evaluating Model's Performance
print('Mean Absolute Error:', mean_absolute_error(y_test, pred))
print('Mean Squared Error:', mean_squared_error(y_test, pred))
print('Mean Root Squared Error:', np.sqrt(mean_squared_error(y_test, pred)))
X_train.shape
X_test.shape
y_test.size
pred.size
iris_df.loc[60]
```

```
d = {'sepal length (cm)' : [5.0],
    'sepal width (cm)' : [3.4],
    'petal length (cm)' : [1.4],
    'petal width (cm)' : [0.3],
    'species' : 0}
test_df = pd.DataFrame(data= d)
test_df
pred = lr.predict(X_test)
print('Predicted Sepal Length (cm):', pred[0])
print('Actual Sepal Length (cm):', 4.6)
import matplotlib.pyplot as plt
_, ax = plt.subplots()
ax.scatter(x = range(0, y_test.size), y=y_test, c = 'blue', label = 'Actual',
alpha = 0.3)
ax.scatter(x = range(0, pred.size), y=pred, c = 'red', label = 'Predicted', alpha
= 0.3)
plt.title('Actual and predicted values')
plt.xlabel('Observations')
plt.ylabel('mpg')
plt.legend()
plt.show()
```

7. WAP to implement K-means clustering

Theory

K-means clustering is one of the simplest and popular unsupervised machine learning algorithms. Typically, unsupervised algorithms make inferences from datasets using only input vectors without referring to known, or labeled, outcomes.
A cluster refers to a collection of data points aggregated together because of certain similarities. You'll define a target number k, which refers to the number of centroids you need in the dataset. A centroid is the imaginary or real location representing the center of the cluster.
Every data point is allocated to each of the clusters through reducing the in-cluster sum of squares.
In other words, the K-means algorithm identifies k number of centroids, and then allocates every data point to the nearest cluster, while keeping the centroids as small as possible.
The 'means' in the K-means refers to averaging of the data; that is, finding the centroid.

Code-

```
import numpy as np
import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
from sklearn.cluster import KMeans
data =
pd.read_csv('https://raw.githubusercontent.com/kastrahl/Data-science/main/cluster
ing.csv')
data.head()
X = data[["LoanAmount","ApplicantIncome"]]
#Visualise data points
plt.scatter(X["ApplicantIncome"],X["LoanAmount"],c='black')
plt.xlabel('AnnualIncome')
plt.ylabel('Loan Amount (In Thousands)')
plt.show()
#number of clusters
K=3
# Select random observation as centroids
Centroids = (X.sample(n=K))
```

```
plt.scatter(X["ApplicantIncome"],X["LoanAmount"],c='black')
plt.scatter(Centroids["ApplicantIncome"],Centroids["LoanAmount"],c='red')
plt.xlabel('AnnualIncome')
plt.ylabel('Loan Amount (In Thousands)')
plt.show()
# -- Assign all the points to the closest cluster centroid
# -- Recompute centroids of newly formed clusters
#  Repeat
diff = 1
j=0
while(diff!=0):
    XD=X
    i=1
    for index1,row_c in Centroids.iterrows():
        ED=[]
        for index2,row_d in XD.iterrows():
            d1=(row_c["ApplicantIncome"]-row_d["ApplicantIncome"])**2
            d2=(row_c["LoanAmount"]-row_d["LoanAmount"])**2
            d=np.sqrt(d1+d2)
            ED.append(d)
        X[i]=ED
        i=i+1

    C=[]
    for index,row in X.iterrows():
        min_dist=row[1]
        pos=1
        for i in range(K):
            if row[i+1] < min_dist:
                min_dist = row[i+1]
                pos=i+1
        C.append(pos)
    X["Cluster"]=C
    Centroids_new =
X.groupby(["Cluster"]).mean()[["LoanAmount","ApplicantIncome"]]
    if j == 0:
        diff=1
```

```
        j=j+1
    else:
        diff = (Centroids_new['LoanAmount'] - Centroids['LoanAmount']).sum() +
(Centroids_new['ApplicantIncome'] - Centroids['ApplicantIncome']).sum()
        print(diff.sum())
    Centroids = X.groupby(["Cluster"]).mean()[["LoanAmount","ApplicantIncome"]]

"""We have initially defined the diff as 1 and inside the while loop, we are
calculating this diff as the difference between the centroids in the previous
iteration and the current iteration.

When this difference is 0, we are stopping the training
"""
color=['blue','green','cyan']
for k in range(K):
    data=X[X["Cluster"]==k+1]
    plt.scatter(data["ApplicantIncome"],data["LoanAmount"],c=color[k])
plt.scatter(Centroids["ApplicantIncome"],Centroids["LoanAmount"],c='red')
plt.xlabel('Income')
plt.ylabel('Loan Amount (In Thousands)')
plt.show()
```

8. WAP to implement market basket analysis using apriori

Theory

Market basket analysis is a data mining technique used by retailers to increase sales by better understanding customer purchasing patterns. It involves analyzing large data sets, such as purchase history, to reveal product groupings, as well as products that are likely to be purchased together.

Apriori Algorithm is a widely-used and well-known Association Rule algorithm and is a popular algorithm used in market basket analysis. It is also considered accurate and overtop AIS and SETM algorithms. It helps to find frequent itemsets in transactions and identifies association rules between these items. The limitation of the Apriori Algorithm is frequent itemset generation. It needs to scan the database many times which leads to increased time and reduce performance as it is a computationally costly step because of a huge database. It uses the concept of Confidence, Support.

Code-

```
# -*- coding: utf-8 -*-
#necessary libraries
import numpy as np
import pandas as pd
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
retdata =
pd.read_csv('https://raw.githubusercontent.com/kastrahl/Data-science/main/MBA.csv
', header = None)
retdata.head()
#Transforming the list into a list of lists, so that each transaction can be
indexed easier
transactions = []
for i in range(0, retdata.shape[0]):
    transactions.append([str(retdata.values[i, j]) for j in range(0, 20)])
print(transactions[0])
! pip install apyori
from apyori import apriori
rules = apriori(transactions, min_support = 0.003, min_confidence = 0.2, min_lift
= 3, min_length = 2)
```

```
# Support: number of transactions containing set of times / total number of
transactions
# Confidence: Should not be too high, as then this wil lead to obvious rules
results = list(rules)
results = pd.DataFrame(results)
results.head(5)
```

9. WAP to implement logistic regression

Theory

Logistic regression is a process of modeling the probability of a discrete outcome given an input variable. The most common logistic regression models a binary outcome; something that can take two values such as true/false, yes/no,
Mathematically, a logistic regression model predicts P(Y=1) as a function of X. It is one of the simplest ML algorithms that can be used for various classification problems such as spam detection, Diabetes prediction, cancer detection etc.

K-nearest neighbors (KNN) algorithm is a type of supervised ML algorithm which can be used for both classification as well as regression predictive problems. However, it is mainly used for classification predictive problems in industry. The following two properties would define KNN well −
Lazy learning algorithm − KNN is a lazy learning algorithm because it does not have a specialized training phase and uses all the data for training while classification.
Non-parametric learning algorithm − KNN is also a non-parametric learning algorithm because it doesn't assume anything about the underlying data.

Code-

```
import pandas as pd
import numpy as np
import os
# used handle some files
import matplotlib.pyplot as plt
# used to visualize the data using graphs
import seaborn as sns
# plotting the chart in a single line
df =
pd.read_csv("https://raw.githubusercontent.com/kastrahl/Data-science/main/Iris.cs
v")
df.head(5)
df = df.drop(columns = ['Id'])
df.head(5)
df.info()
df['Species'].value_counts()
df.isnull().sum()
```

```
df['SepalLengthCm'].hist()
df['SepalWidthCm'].hist()
df['PetalLengthCm'].hist()
df['PetalWidthCm'].hist()
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['Species'] = le.fit_transform(df['Species'])
df.head(100)
from sklearn.model_selection import train_test_split, KFold
X = df.drop(columns = ['Species'])
Y = df['Species']
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25)
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train, Y_train)
print("Accuracy: ", model.score(X_test, Y_test) * 100)

"""KNN """
from sklearn.preprocessing import Normalizer
from sklearn.metrics import accuracy_score
from sklearn.neighbors import KNeighborsClassifier
from collections import Counter
iris_df=df
iris_df.head()
x= iris_df.iloc[:, :-1]
y= iris_df.iloc[:, -1]
# split the data into train and test sets
x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.2,shuffle=
True, #shuffle the data to avoid biasrandom_state= 0)
x_train= np.asarray(x_train)
y_train= np.asarray(y_train)
x_test= np.asarray(x_test)
y_test= np.asarray(y_test)
```