Document Scope: Python/Deeplearning Lab assignment

CSEE5590 – Python/ DeepLearning Programming

LAB ASSIGNMENT – 1 REPORT

Team #5

Name: Saitejaswi Koppuravuri

Class ID: 16

Name: Kartheek Katta

Class ID: 12

Name: Aanchal Tiwari

Class ID: 38

Objective:

The main objective of this lab tutorial is to get the basic understanding of python and some concepts of machine learning algorithms i.e., classification algorithms like SVM, Naive Bayes, K-Nearest Neighbors, Regression algorithms like multiple linear regression, Clustering techniques like KMeans Clustering, along with their corresponding metrics.

Technologies/IDE's used:

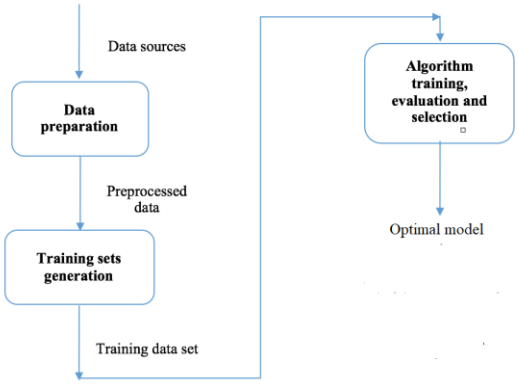
- Python 3.7
- Pycharm IDE

Libraries used:

- Numpy
- Pandas
- Scikit learn library

Workflow:

The workflow of the any machine learning algorithm in this lab assignment is as follows:



- As per the workflow, initially, a dataset is taken, pre-processing on the dataset like calculating the null values, replacing the obtained null values, correlation in between features are performed.
- Next step is splitting the whole dataset into two different parts namely training and testing, where the model is trained based on the trained dataset and it is being validated on the test data.
- Now, the model is being fit, and corresponding metrics are calculated.

Program 1:

Suppose you have a list of tuples as follows:

[(‘John’, (‘Physics’, 80)) , (‘ Daniel’, (‘Science’, 90)), (‘John’, (‘Science’, 95)), (‘Mark’,(‘Maths’, 100)), (‘Daniel’, (’History’, 75)), (‘Mark’, (‘Social’, 95))]

Create a dictionary with keys as names and values as list of (subjects, marks)in sorted order.

{John : [(‘Physics’, 80), (‘Science’, 95)]Daniel : [(’History’, 75), (‘Science’, 90)]Mark : [(‘Maths’, 100), (‘Social’, 95)]}

Python code:

```
student_list = [("John", ("Physics", 80)), ("Daniel", ("Science", 90)), ("John",("Science", 95)), ("Mark",("Maths", 100)), ("Daniel", ("History",
student_dict = {}

for student in student_list:
    if student[0] not in student_dict:
        student_dict[student[0]] = list()
        student_dict[student[0]].append(student[1])
    else:
        student_dict[student[0]].append(student[1])

for student in student_dict:
    print(student,":", student_dict[student])
```

- Initially, the given data in the form of tuples is taken into a list and initializing an empty dictionary.
- Now, iterating through the list adding the items into the dictionary in the desired way and finally printing the dictionary.

Output:

```
John : [('Physics', 80), ('Science', 95)]
Mark : [('Maths', 100), ('Social', 95)]
Daniel : [('Science', 90), ('History', 75)]

Process finished with exit code 0
```

Program 2:

Given a string, find the longest substrings without repeating characters along with the length as a tuple Input:

"pwwkew" Output: (wke,3), (kew,3)

Document Scope: Python/Deeplearning Lab assignment

Python code:

```
temp = ""
dict = {}
for j in range(len(str)):
    for i in range(j,len(str)):
        if not(str[i] in temp):
            temp += str[i]
        else:
            dict[temp] = len(temp)
            temp = ''
            break
max_val = max(dict.values())
list1=[]
for key, val in dict.items():
    if max_val == val:
        list1.append((key, val))
```

- Initially, create a temp string where we store all the non-repeating characters.
- Now, iterate through the length of the input and check whether the particular character is in the temp and if not add that character.
- And finally print the longest substrings.

Output:

```
[('kew', 3), ('wke', 3)]

Process finished with exit code 0
```

Program 3:

Write a python program to create any one of the following management systems.

1. Airline Booking Reservation System (e.g. classes Flight, Person, Employee, Passenger etc.)

2. Library Management System(e.g: Student, Book, Faculty, Department etc.)

- I have created the Library management system, which has 5 different classes namely Person, Student, Librarian, Book, Borrow_book.
- Person is the main class and the classes Student and Librarian have inherited (single inheritance) the Person class.
- Borrow_Book class implements multiple inheritance with base class Student, Book.
- Declared private data member StudentCount in student class to count number of student objects created.

Document Scope: Python/Deeplearning Lab assignment

- Used super call in class Librarian to initialize Person class object.
- Used a private member __numBooks for keeping the track of books.

Python code:

The main class(Person class) is as follows:

```
class Person:
    def __init__(self,name,email):
        self.name = name
        self.email = email

    def display(self):
        print("Name: ", self.name)
        print("Email: ", self.email)
```

Concept of inheritance by Student class:

```
# Inheritance concept where student is inheriting the Person class


class Student(Person):
    StudentCount = 0

    def __init__(self,name,email,student_id):
        Person.__init__(self,name,email)
        self.student_id = student_id
        Student.StudentCount +=1
```

Super call in the librarian Class:

```
class Librarian(Person):
    StudentCount = 0

    def __init__(self,name,email,employee_id):
        # super call where Librarian class is inheriting the Person class
        super().__init__(name,email)
        self.employee_id = employee_id
```

Creating a private number:

Document Scope: Python/Deeplearning Lab assignment

```
class Book():
    __numBooks = 0   # private member
    def __init__(self,book_name,author,book_id):
        self.book_name = book_name
        self.author = author
        self.book_id = book_id
        Book.__numBooks += 1   # keeps track of which student or staff has book checked
```

Multiple inheritance concept for Borrow_book class:

```
class Borrow_Book(Student,Book):

    def __init__(self,name,email,student_id,book_name,author,book_id):
        Student.__init__(self,name,email,student_id)
        Book.__init__(self,book_name,author,book_id)

    def display(self):
        print("Borrowed Book Details:")
        Student.display(self)
        Book.display(self)
```

Creating instances:

```
# creating instances of all classes
Records = []
Records.append(Student('xyz','xyz@gmail.com',123))
Records.append(Librarian('abc','xyz@gmail.com',789))
Records.append(Book('davinci code','leo',123456))
Records.append(Borrow_Book('def','pqr@gmail.com',456,'wings of fire','kalam',67890))
```

Output:

```
Student Details:
Name:  xyz
Email:  xyz@gmail.com
Student Id:  123


Employee Details:
Name:  abc
Email:  xyz@gmail.com
Employee Id:  789


Book Details
Book_Name:  davinci code
Author:  leo
Book_ID:  123456


Borrowed Book Details:
Student Details:
Name:  def
Email:  pqr@gmail.com
Student Id:  456
Book Details
Book_Name:  wings of fire
Author:  kalam
Book_ID:  67890


Total Number of Students: 2

Process finished with exit code 0
```

Program 4:

Create Multiple Regression by choosing a dataset of your choice (again before evaluating, clean the data set with the EDA learned in the class). Evaluate the model using RMSE and R2 and also report if you saw any improvement before and after the EDA.

The dataset I have choosen is part of sklearn.datasets.load_diabetes
Dataset link: https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_diabetes.html#sklearn.datasets.load_diabetes

Python code:

Importing the necessary libraries:

```
import numpy as np
import pandas as pd
from sklearn.datasets import load_diabetes
import matplotlib.pyplot as plt

diabetes = load_diabetes()
dataset = pd.DataFrame(data=np.c_[diabetes['data'], diabetes['target']],
                       columns=diabetes['feature_names'] + ['target'])
```

Creating the features and the target:

```
x = dataset.drop(['target'], axis=1)
y = np.log(dataset.target)
```

So, there is no null value.

- encoding the categorical features From the result above, we do not have to encode any data in this dataset to numeric.
- removing the features not correlated to the target class

The first five features are the most positively correlated with target and the next five are the most negatively correlated.

Correlating the features:

```
corr = dataset.corr()
print(corr['target'].sort_values(ascending=False)[:6], '\n')
print(corr['target'].sort_values(ascending=False)[-6:])
```

Output:

```
target    1.000000
bmi       0.586450
s5        0.565883
bp        0.441484
s4        0.430453
s6        0.382483
Name: target, dtype: float64

s6     0.382483
s1     0.212022
age    0.187889
s2     0.174054
sex    0.043062
s3    -0.394789
```

- Based on this result, the festures bmi, s5, bp, s4, s6, s3 should be considered more (cause their score is more than 0.3)

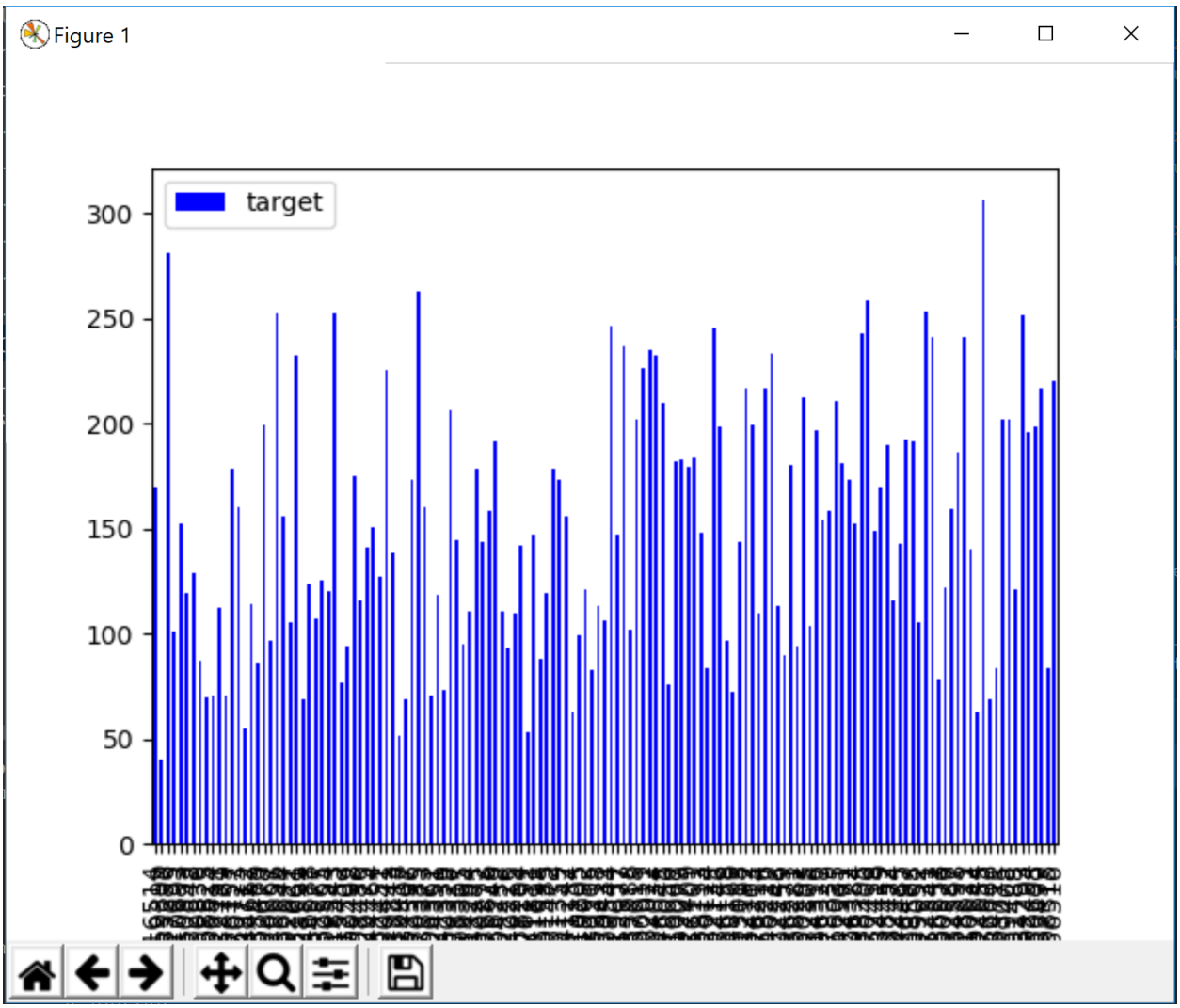
Pivot table for the for the s1, age, s2, sex. Because we are still thinking whether keep them in to the model.

Creating the pivot plots:

Document Scope: Python/Deeplearning Lab assignment

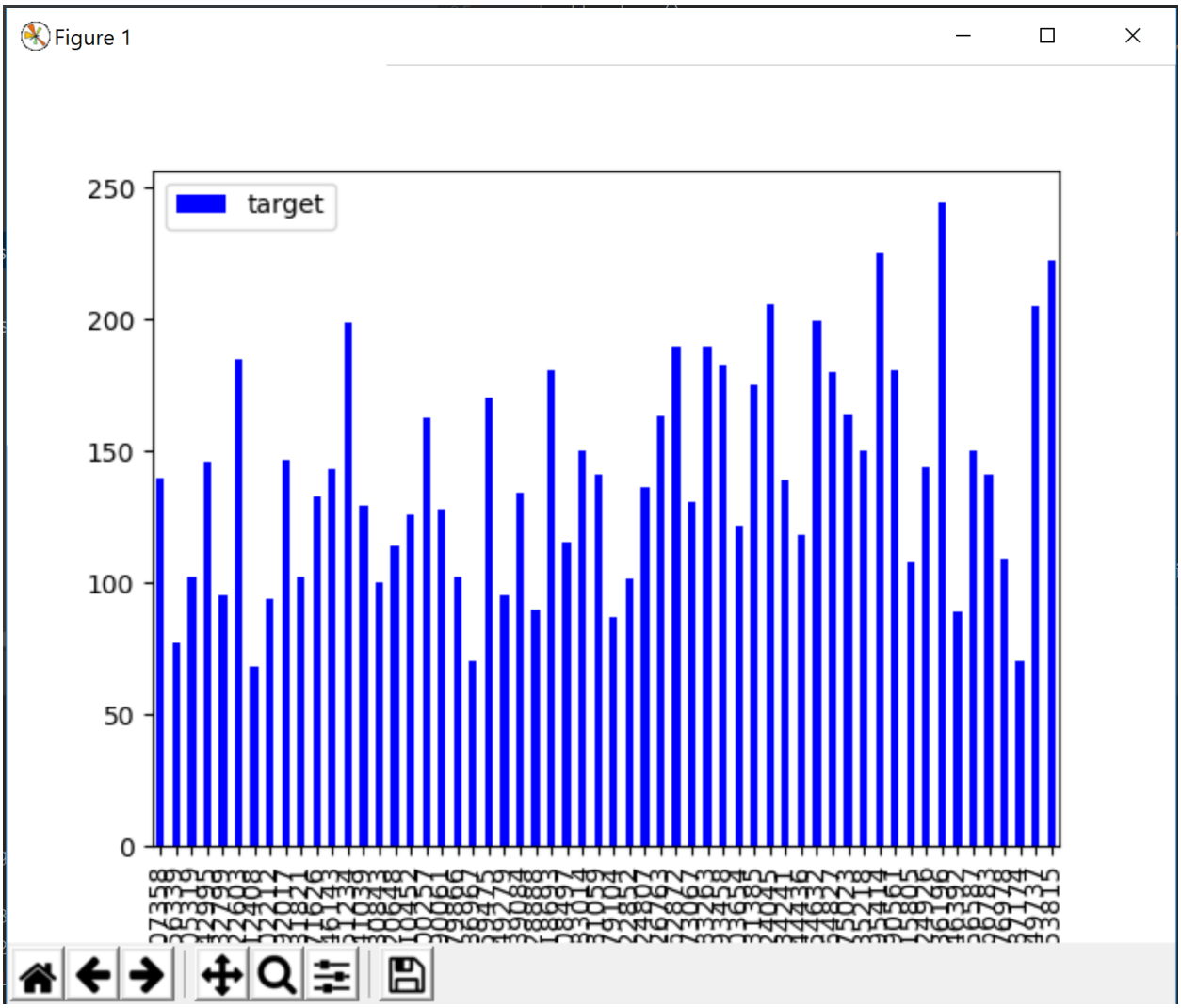
```
quality_pivot = dataset.pivot_table(index='s1', values='target', aggfunc=np.median)
quality_pivot.plot(kind='bar', color='blue')
plt.show()
quality_pivot = dataset.pivot_table(index='age', values='target', aggfunc=np.median)
quality_pivot.plot(kind='bar', color='blue')
plt.show()
quality_pivot = dataset.pivot_table(index='s2', values='target', aggfunc=np.median)
quality_pivot.plot(kind='bar', color='blue')
plt.show()
quality_pivot = dataset.pivot_table(index='sex', values='target', aggfunc=np.median)
quality_pivot.plot(kind='bar', color='blue')
plt.show()
```

Output of pivot plots: s1

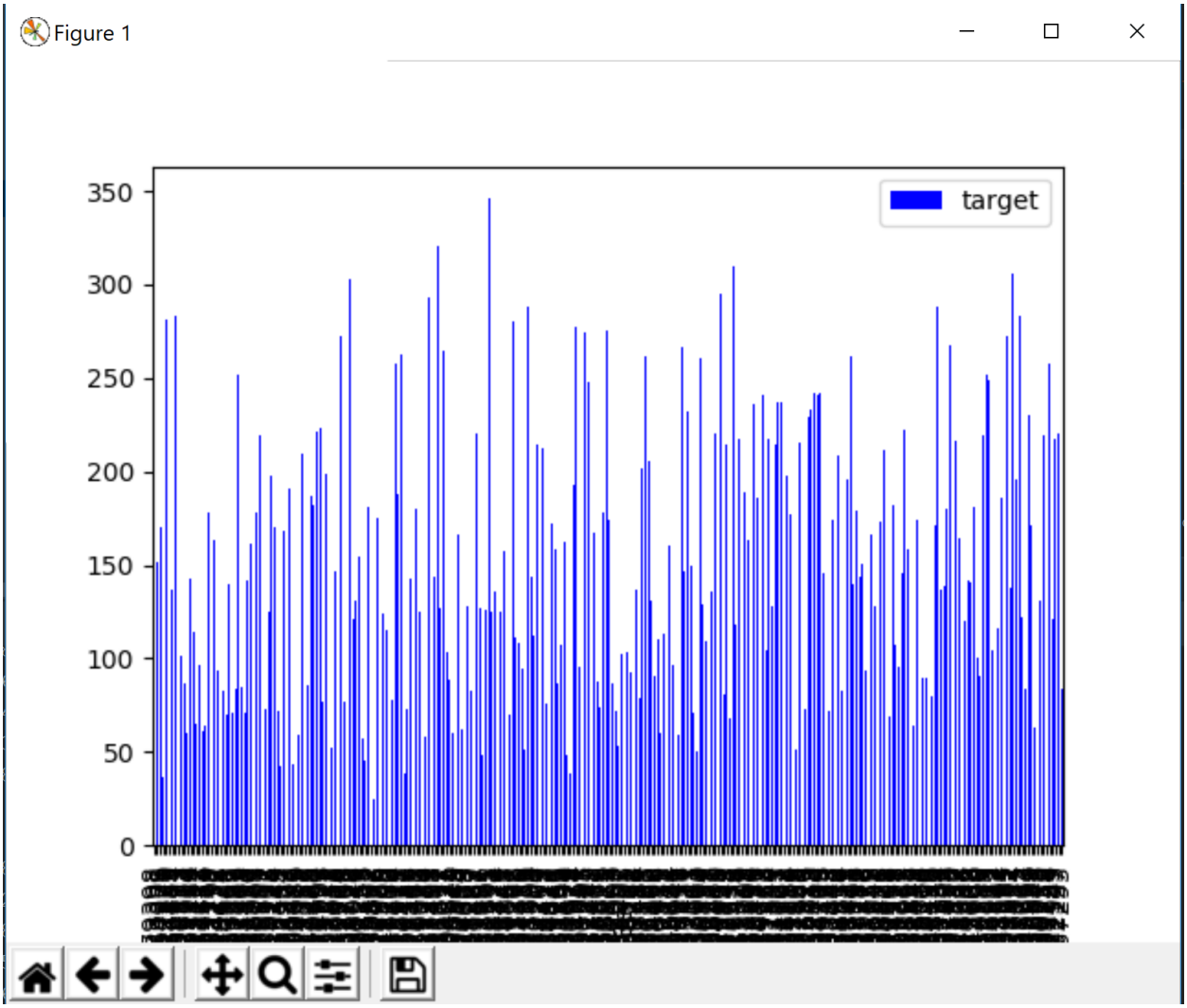


age

Document Scope: Python/Deeplearning Lab assignment



s2



From the plots, these features are not correlated with target, so we should take them out of the dataset.

Splitting the data in terms of train and split:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, y, random_state=42, test_size=.33)
```

Fitting the model and calculating the scores:

```
from sklearn import linear_model
lr1 = linear_model.LinearRegression()
model = lr1.fit(X_train, y_train)

print('r2 is: ', model.score(X_test, y_test))
prediction = model.predict(X_test)
from sklearn.metrics import mean_squared_error
print('rmse: ', mean_squared_error(y_test, prediction))
```

Output before eliminating the features:

```
r2 is:  0.466670995368768
rmse:  0.157748723592490 6
```

Output after eliminating the features:

```
r2 is:  0.4803686246810386 4
rmse:  0.153697221571242 3

Process finished with exit code 0
```

We can observe a slight increase in the score as because we are eliminating the data that are less correlated to the target variable.

Program 5:

Pick any dataset from the dataset sheet in the class sheet or online which includes both numeric and non-numeric features

a. Perform exploratory data analysis on the data set (like Handling null values, removing the features not correlated to the target class, encoding the categorical features, ...)

b. Apply the three classification algorithms Naïve Baye's, SVM and KNN on the chosen data set and report which classifier gives better result.

Document Scope: Python/Deeplearning Lab assignment

Dataset: Car Evaluation Data Set from UC Irvine Machine Learning Repository

Dataset Link: http://archive.ics.uci.edu/ml/datasets/Car+Evaluation

Features:

buying - v-high, high, med, low

maint - v-high, high, med, low

doors - 2, 3, 4, 5-more

persons - 2, 4, more

lug_boot - small, med, big

safety - low, med, high

Target variable:

class values - unacc, acc, good, vgood

Python code:

Importing required libraries

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns



dataset = pd.read_csv('car.txt', sep=",", header=None)
dataset.columns = ["buying", "maint", "doors", "persons", "lugBoot", "safety", "classValue"]
# dataset.info()
print(dataset.isnull().sum())
```

Output:

```
buying       0
maint        0
doors        0
persons      0
lugBoot      0
safety       0
classValue   0
```

Label encoder:

So, there is non-null value in the dataset need to be handle.

- encoding the categorical features There is all object type data in dataset, even for doors and person, which are already numbers in dataset.
- So, I transform them into numeric features.
- Based on the dataset features, I assign the number to them like: number 0,1,2,3 to low, med, high, v-high in feature buying number 0,1,2,3 to low, med, high, v-high in feature maint number 6 to 5-more in feature doors number 6 to more in feature persons number 0,1,2 to small, med, big in feature lug_boot number 0,1,2 to low, med, high in feature safety And target number 0,1,2,3 to unacc, acc, good, vgood in feature class value

```
# Transforming and engineering non-numeric features
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
Num_dataset = dataset
```

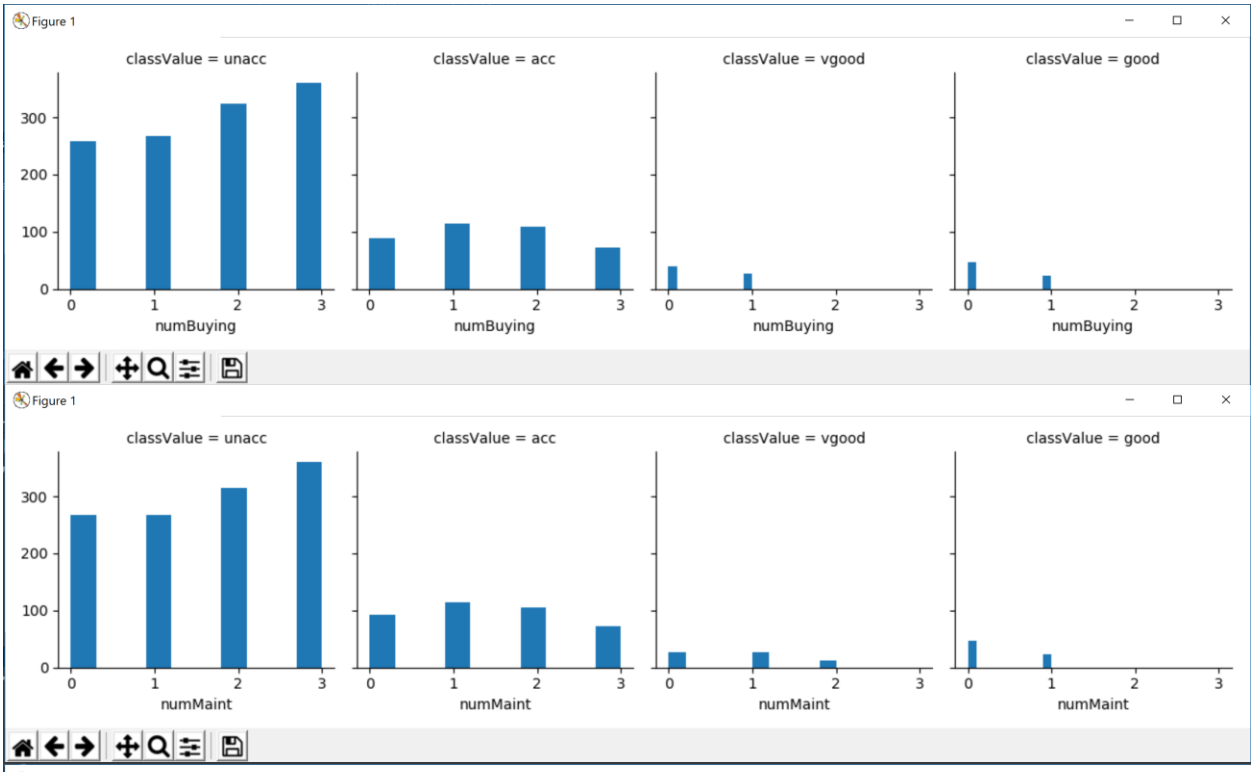
Correlation:

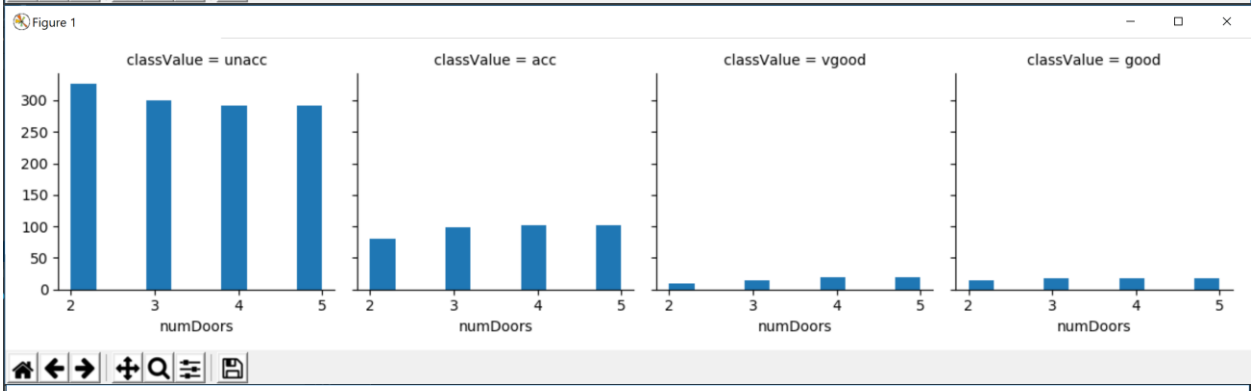
- removing the features not correlated to the target class
- In this part, I calculated the correlation betweem each feature and target.(numeric data)

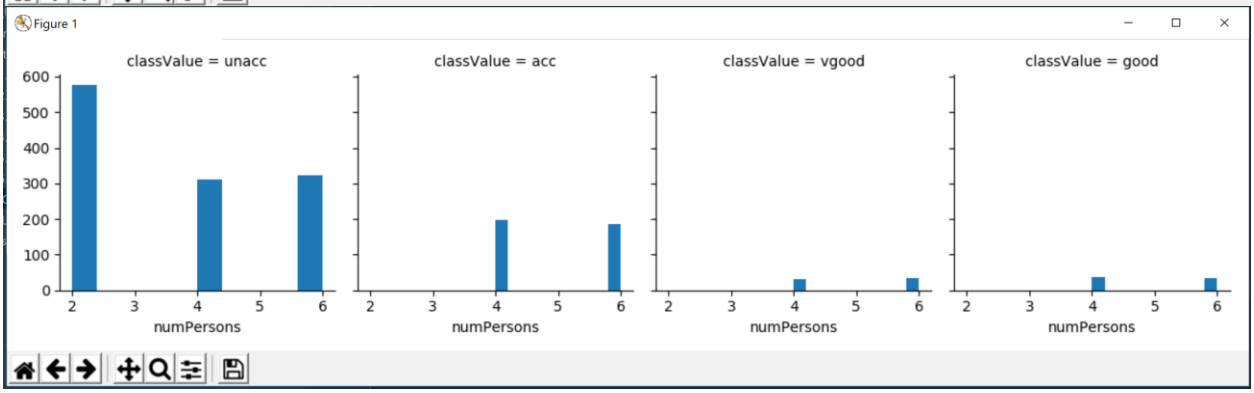
```
buy_corr = Num_dataset[['buying', 'numValue']].groupby(['buying'], as_index=False).mean().sort_values(by='numValue', ascending=False)
maint_corr = Num_dataset[["maint", "numValue"]].groupby(['maint'], as_index=False).mean().sort_values(by='numValue', ascending=False)
door_corr = Num_dataset[["doors", "numValue"]].groupby(['doors'], as_index=False).mean().sort_values(by='numValue', ascending=False)
person_corr = Num_dataset[["persons", "numValue"]].groupby(['persons'], as_index=False).mean().sort_values(by='numValue', ascending=False)
boot_corr = Num_dataset[["lugBoot", "numValue"]].groupby(['lugBoot'], as_index=False).mean().sort_values(by='numValue', ascending=False)
safe_corr = Num_dataset[["safety", "numValue"]].groupby(['safety'], as_index=False).mean().sort_values(by='numValue', ascending=False)
```

Output plots of correlating features:

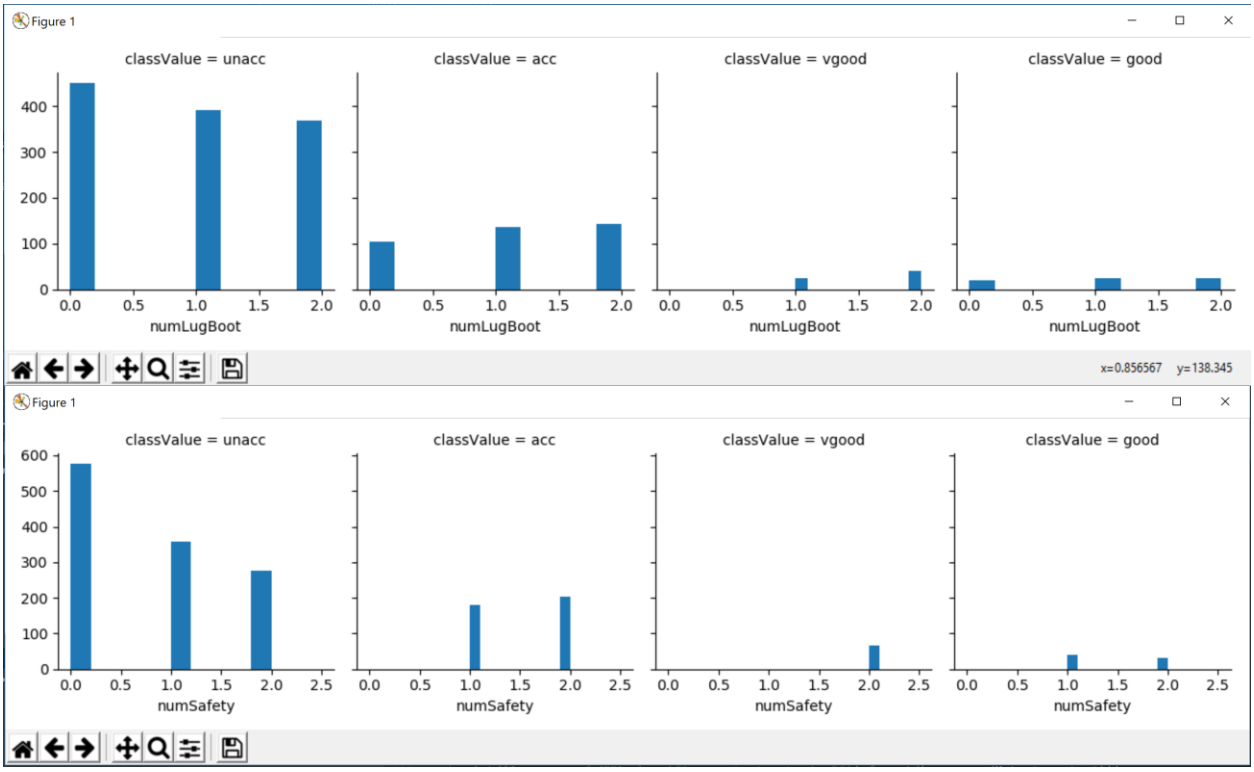
Document Scope: Python/Deeplearning Lab assignment







Document Scope: Python/Deeplearning Lab assignment



Creating features and target variable:

```
data_train = Num_dataset.drop(["buying", "maint", "doors", "persons", "lugBoot", "safety", "classValue"], axis=1)
x = data_train.drop("numValue", axis=1)
y = dataset.classValue
```

Gaussian NB Model:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, y, random_state=42, test_size=.33)

# NB
from sklearn.naive_bayes import GaussianNB
GNB = GaussianNB()
GNB.fit(X_train, y_train)
print("\n----------GNB----------")
# GaussianNB(priors=None, var_smoothing=1e-09)
print('Accuracy of Naive Bayes GaussianNB on training set: {:.2f}'.format(GNB.score(X_train, y_train)))
# Evaluate the model on testing part
print('Accuracy of Naive Bayes GaussianNB on test set: {:.2f}'.format(GNB.score(X_test, y_test)))
```

SVM Model:

```
# SVM
from sklearn.svm import SVC
svm = SVC(kernel='rbf')
svm.fit(X_train, y_train)
print("\n----------SVM----------")
print('Accuracy of SVM classifier on training set: {:.2f}'.format(svm.score(X_train, y_train)))
# test data set acc
print('Accuracy of SVM classifier on test set: {:.2f}'.format(svm.score(X_test, y_test)))
```

KNN Model:

```
# KNN
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=2)
knn.fit(X_train, y_train)
print("\n----------KNN----------")
print('Accuracy of KNN classifier on training set: {:.2f}'.format(knn.score(X_train, y_train)))
# test data set acc
print('Accuracy of KNN classifier on test set: {:.2f}'.format(knn.score(X_test, y_test)))
```

Output scores of all the models:

```
----------GNB----------
Accuracy of Naive Bayes GaussianNB on training set: 0.71
Accuracy of Naive Bayes GaussianNB on test set: 0.70
C:\Users\Koppu\PycharmProjects\Lab\venv\lib\site-packages\sklearn\svm\ba
  "avoid this warning.", FutureWarning)

----------SVM----------
Accuracy of SVM classifier on training set: 0.98
Accuracy of SVM classifier on test set: 0.95

----------KNN----------
Accuracy of KNN classifier on training set: 0.92
Accuracy of KNN classifier on test set: 0.84

Process finished with exit code 0
```

From the result, we can see that the SVM model is the best with highest accuracy for both training dataset and testing dataset!

Program 6:

Choose any dataset of your choice. Apply K-means on the dataset and visualize the clusters using matplotlib or seaborn.

a. Report which K is the best using the elbow method.

b. Evaluate with silhouette score or other scores relevant for unsupervised approaches (before applying clustering clean the data set with the EDA learned in the class)

Python code:

Importing libraries:

```
import pandas as pd
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(style="white", color_codes=True)
import warnings
warnings.filterwarnings("ignore")

data = pd.read_csv('Mall_Customers.csv')


print(data["Spending_Score"].value_counts())
```

Checking nulls:

```
nulls = pd.DataFrame(data.isnull().sum().sort_values(ascending=False)[:25])
nulls.columns = ['Null Count']
nulls.index.name = 'Feature'
print(nulls)
```

Output:

```
Null Count
Feature
Spending_Score                0
Annual Income (k$)            0
Age                           0
Gender                        0
CustomerID                    0
(200, 2) (200,)
```

Creating features and target:

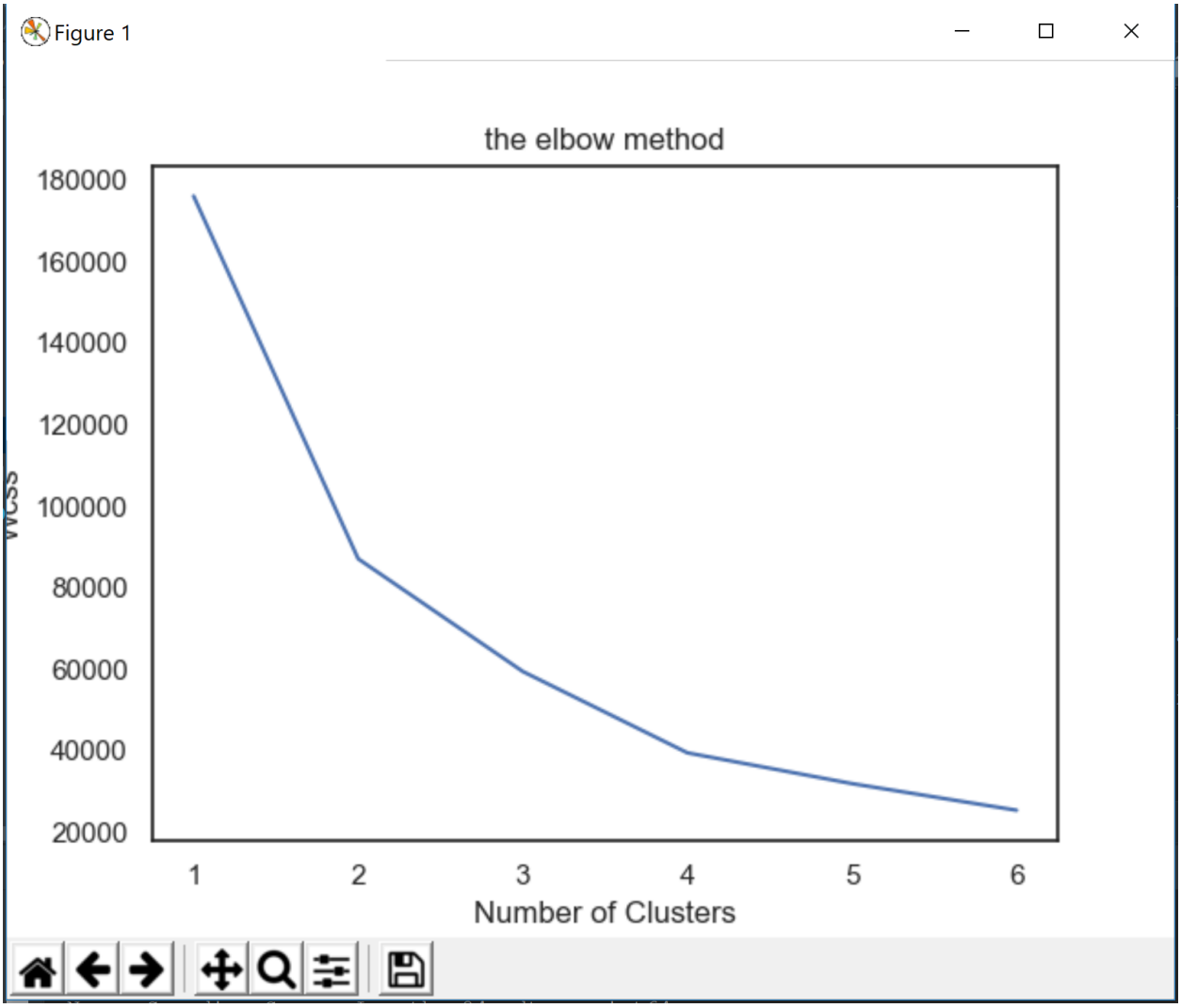
```
x = data.iloc[:,2:-1]
y = data.iloc[:,-1]
print(x.shape,y.shape)
```

Elbow Method:

```
#elbow method to know the number of clusters

wcss = []
for i in range(1,7):
    kmeans = KMeans(n_clusters=i,init='k-means++',max_iter=300,n_init=10,random_state=0)
    kmeans.fit(x)
    wcss.append(kmeans.inertia_)
print(wcss)
plt.plot(range(1,7),wcss)
plt.title('the elbow method')
plt.xlabel('Number of Clusters')
plt.ylabel('Wcss')
plt.show()
```

Output:

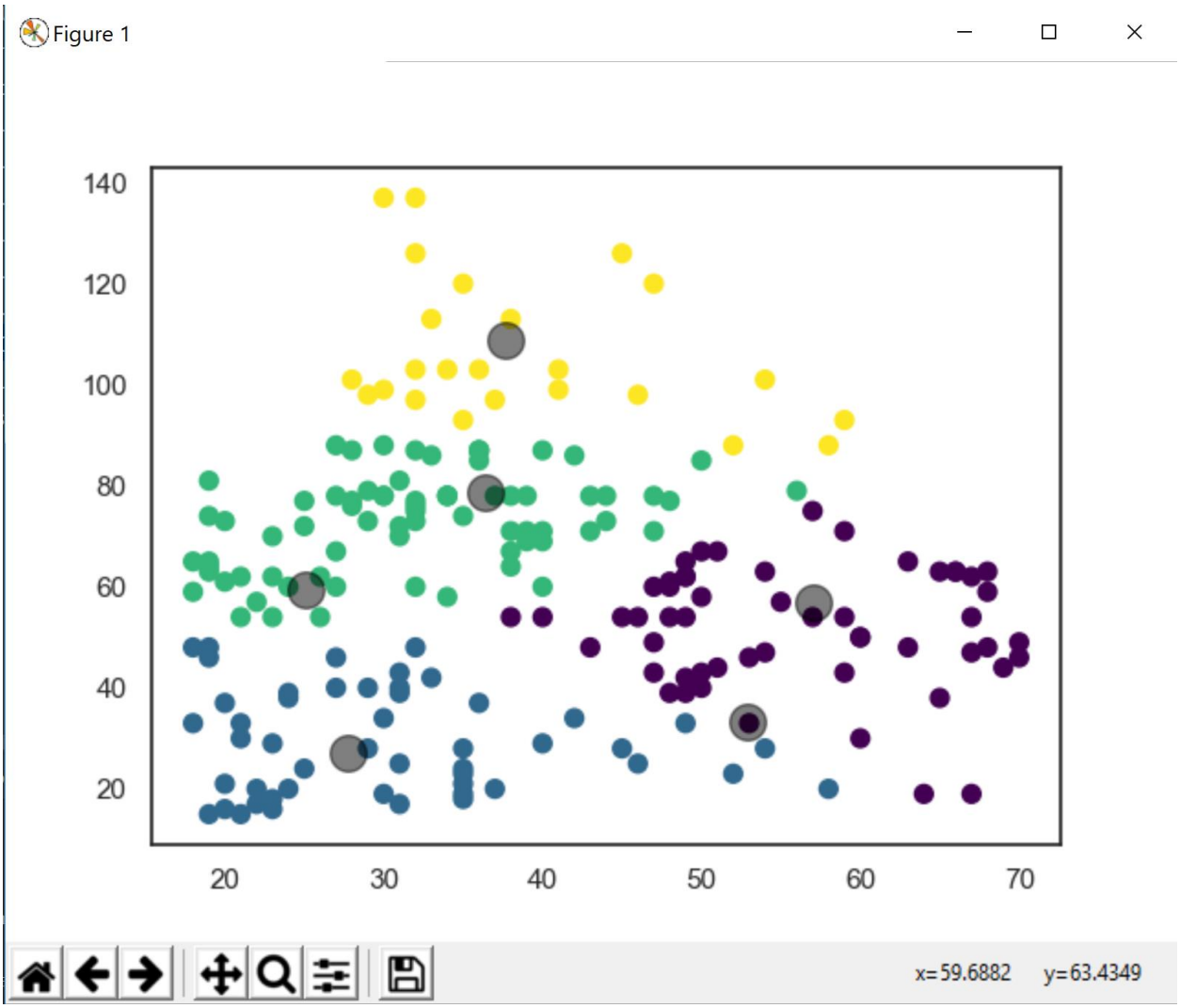


Performing PCA:

```
# standardization
pca = PCA(2)
x_pca = pca.fit_transform(x)
df2 = pd.DataFrame(data=x_pca)
finaldf = pd.concat([df2,data[['Spending_Score']]],axis=1)
print(finaldf)
```

Output of clustering:

Document Scope: Python/Deeplearning Lab assignment



Before PCA result:

After PCA score and result:

```
188   42.418440    2.540092                17
189   42.464409   -2.459697                85
190   42.482796   -4.459613                23
191   42.501183   -6.459528                69
192   52.491567   -5.367633                 8
193   52.445598   -0.367844                91
194   59.362559    8.696132                16
195   59.472884   -3.303361                79
196   65.380693    6.751379                28
197   65.500211   -6.248072                74
198   76.499747   -6.146941                18
199   76.518134   -8.146856                83

[200 rows x 3 columns]
0.43295184273333087
```