

Python for Deep Learning

LAB 1 Documentation

Lab ID: 7

Avinash Ganguri (6)

Nikhil Kantipudi (10)

Dileep Reddy Peddakam (19)

Introduction

Implemented several Machine learning techniques such as Classification, Regression, Natural Language Processing, dimensionality reduction and dealt with several python libraries such as Scikit learn and nltk, wordnet etc in the following 8 programs

Objective

To get optimized output for different programs based on ML techniques such as Classification, Natural Language Processing and Regression, etc...

Approaches/Methods

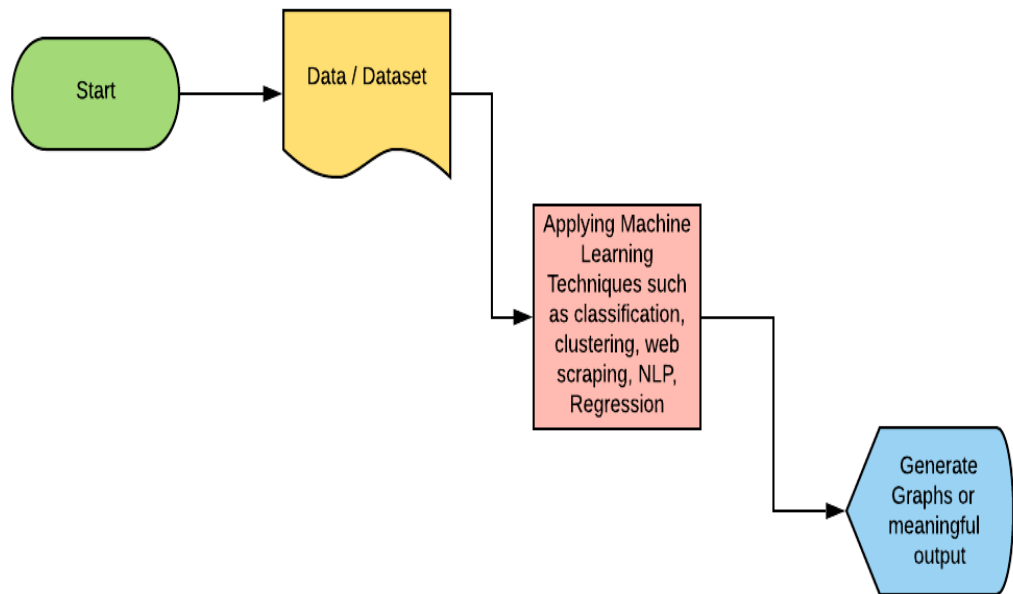
We used approaches or methods such as,

- Object oriented programming
- Web Scraping
- Data Analysis
- Classification algorithms such as Naïve Bayes, SVM and KNN
- Clustering algorithms like K-means, plotting Elbow graph
- Calculate silhouette score
- Applications of NLP
- Multiple Regression

Datasets

- For webscraping : <https://catalog.umkc.edu/course-offerings/graduate/comp-sci/>
- CC.CSV
- GLASS.CSV
- NPL_INPUT.TXT
- TRAIN.CSV
- WINEQUALITY-RED.CSV

Workflow



Evaluation of Programs

1)



The screenshot shows a Jupyter Notebook titled 'lab1_1.ipynb'. The code in the cell is as follows:

```
[2] from itertools import combinations

n = int(input("Enter array size:"))
print ("Enter values:")
array = []
for i in range(0, n):
    val = int(input())
    array.append(val)
for i in range(1,n+1):
    print(list(combinations(array, i)))
```

The output of the code is:

```
Enter array size:5
Enter values:
6
7
8
9
7
[(6,), (7,), (8,), (9,), (7,)]
[(6, 7), (6, 8), (6, 9), (6, 7), (7, 8), (7, 9), (7, 7), (8, 9), (8, 7), (9, 7)]
[(6, 7, 8), (6, 7, 9), (6, 7, 7), (6, 8, 9), (6, 8, 7), (6, 9, 7), (7, 8, 9), (7, 8, 7), (7, 9, 7), (8, 9, 7)]
[(6, 7, 8, 9), (6, 7, 8, 7), (6, 7, 9, 7), (6, 8, 9, 7), (7, 8, 9, 7)]
[(6, 7, 8, 9, 7)]
```

In this program we take values from the user which might contains the duplicate values. By using combinations on the arrays we displayed the subsets of the given numbers. Since we are using the combinations it doesn't display the null values.

2)



The screenshot shows a Jupyter Notebook titled 'lab1_2.ipynb'. The code in the cell is as follows:

```
[1] a = {'hello': 1, 'ram': 2, 'there': 3}
b = {'who': 6, 'he': 4, 'at': 5}
b.update(a)
# a and b are two dict
# concatenating two dict

for key, value in sorted(b.items(), key=lambda item: item[1]):
    print("%s: %s" % (key, value))
```

The output of the code is:

```
hello: 1
ram: 2
there: 3
he: 4
at: 5
who: 6
```

In this program we declared two dictionaries. Update method is used to merge the dictionaries. By using the lamda function and the sorted method we sorted these dictionaries based on the values.

3)



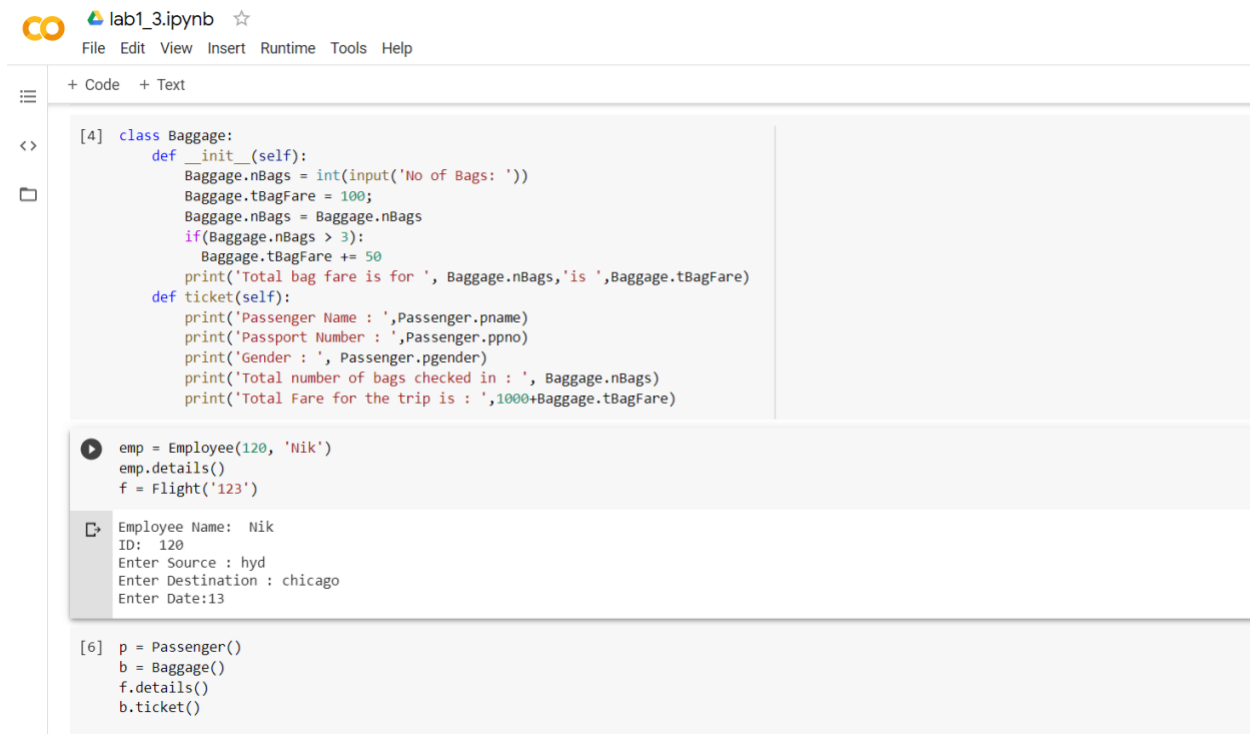
The screenshot shows a Jupyter Notebook titled 'lab1_3.ipynb'. The interface includes a top bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', 'Help', and 'All changes saved'. Below the top bar is a sidebar with icons for 'Code' and 'Text'. The main area displays three code cells:

```
[1] class Flight:
    def __init__(self, fno):
        Flight.source = input('Enter Source : ')
        Flight.dest = input('Enter Destination : ')
        Flight.date=input('Enter Date:')
        self.fno = fno
    def details(self):
        print('Flight Date : ', Flight.date)
        print('Flight Number : ', self.fno)
        print(Flight.source, '-->', Flight.dest)

[2] class Employee(Flight): #inheritance
    def __init__(self, eid, ename):
        self.eid = eid
        self.ename = ename
    def details(self): #Method overloading
        print("Employee Name: ", self.ename)
        print('ID: ', self.eid)

[3] class Passenger:
    def __init__(self):
        Passenger.pname = input('Enter Name : ')
        Passenger.ppno = input('Enter passport No: ')
        Passenger.pgender = input('Enter Gender : ')
```

In this program, flight class accepts the input from the user and details method which displays the values and the second class which is employee will display the employee details by using the inheritance and method overloading. The passenger will take the input from the user.



lab1_3.ipynb

File Edit View Insert Runtime Tools Help

+ Code + Text

```
[4] class Baggage:
    def __init__(self):
        Baggage.nBags = int(input('No of Bags: '))
        Baggage.tBagFare = 100;
        Baggage.nBags = Baggage.nBags
        if(Baggage.nBags > 3):
            Baggage.tBagFare += 50
        print('Total bag fare is for ', Baggage.nBags,'is ',Baggage.tBagFare)
    def ticket(self):
        print('Passenger Name : ',Passenger.pname)
        print('Passport Number : ',Passenger.ppno)
        print('Gender : ', Passenger.pgender)
        print('Total number of bags checked in : ', Baggage.nBags)
        print('Total Fare for the trip is : ',1000+Baggage.tBagFare)
```

```
emp = Employee(120, 'Nik')
emp.details()
f = Flight('123')
```

```
Employee Name: Nik
ID: 120
Enter Source : hyd
Enter Destination : chicago
Enter Date:13
```

```
[6] p = Passenger()
b = Baggage()
f.details()
b.ticket()
```

Here Baggage class take the input of number of bags so that it will calculate the fare



lab1_3.ipynb

File Edit View Insert Runtime Tools Help [All changes saved](#)

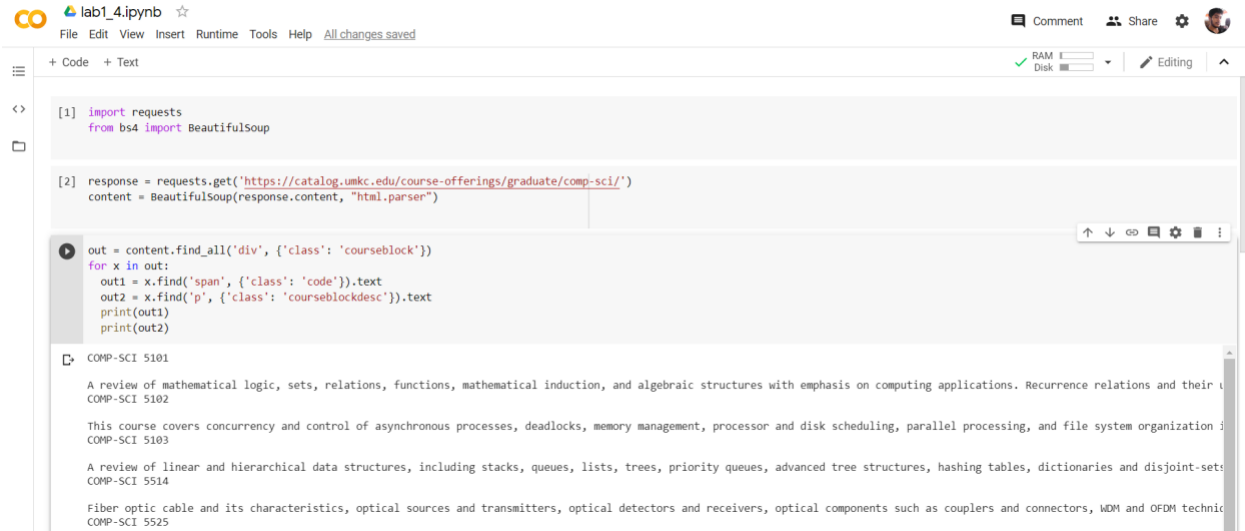
+ Code + Text

```
p = Passenger()
b = Baggage()
f.details()
b.ticket()
```

```
Enter Name : dil
Enter passport No: 457
Enter Gender : m
No of Bags: 3
Total bag fare is for 3 is 100
Flight Date : 13
Flight Number : 123
hyd --> chicago
Passenger Name : dil
Passport Number : 457
Gender : m
Total number of bags checked in : 3
Total Fare for the trip is : 1100
```

Finally ticket class takes all the details and it will give the fare of the trip.

4)



```

lab1_4.ipynb
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[1] import requests
    from bs4 import BeautifulSoup

[2] response = requests.get('https://catalog.umkc.edu/course-offerings/graduate/comp-sci/')
    content = BeautifulSoup(response.content, "html.parser")

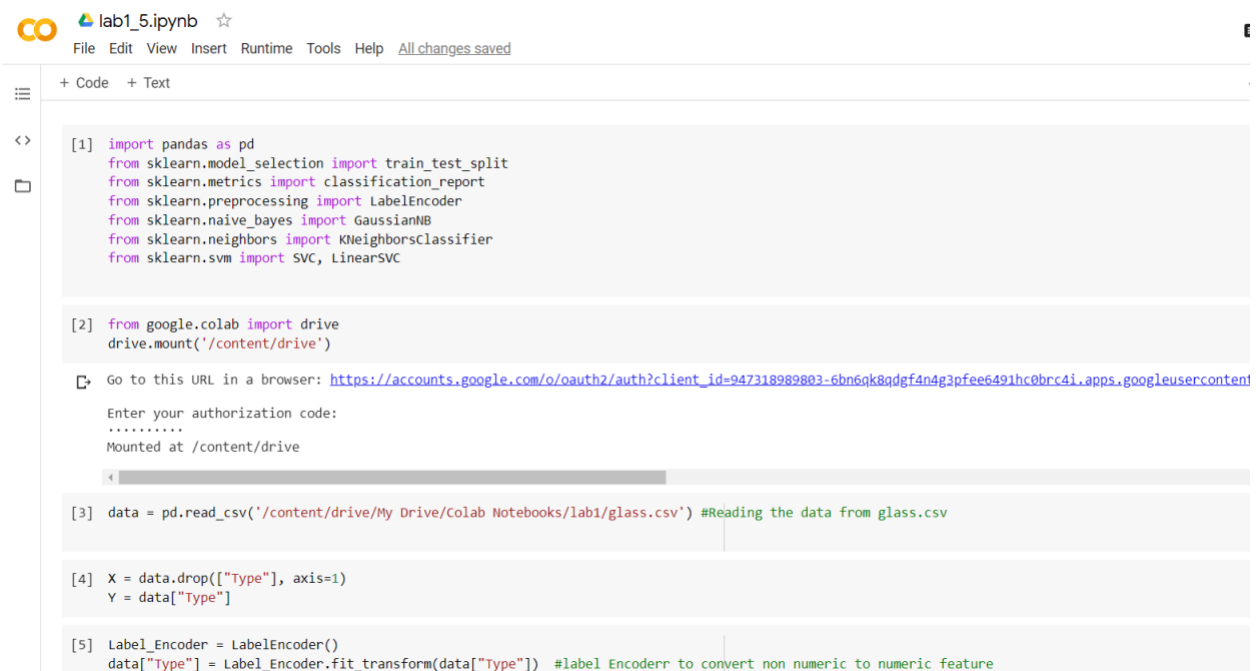
[3] out = content.find_all('div', {'class': 'courseblock'})
    for x in out:
        out1 = x.find('span', {'class': 'code'}).text
        out2 = x.find('p', {'class': 'courseblockdesc'}).text
        print(out1)
        print(out2)

COMP-SCI 5101
A review of mathematical logic, sets, relations, functions, mathematical induction, and algebraic structures with emphasis on computing applications. Recurrence relations and their applications.
COMP-SCI 5102
This course covers concurrency and control of asynchronous processes, deadlocks, memory management, processor and disk scheduling, parallel processing, and file system organization.
COMP-SCI 5103
A review of linear and hierarchical data structures, including stacks, queues, lists, trees, priority queues, advanced tree structures, hashing tables, dictionaries and disjoint-sets.
COMP-SCI 5514
Fiber optic cable and its characteristics, optical sources and transmitters, optical detectors and receivers, optical components such as couplers and connectors, WDM and OFDM techniques.
COMP-SCI 5525

```

In this program we get the html link with the help of BeautifulSoup package and convert that into text we give div tags with class courseblock then in loop to get course names and to display text without tags we use .text and the output is displayed.

5)



```

lab1_5.ipynb
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[1] import pandas as pd
    from sklearn.model_selection import train_test_split
    from sklearn.metrics import classification_report
    from sklearn.preprocessing import LabelEncoder
    from sklearn.naive_bayes import GaussianNB
    from sklearn.neighbors import KNeighborsClassifier
    from sklearn.svm import SVC, LinearSVC

[2] from google.colab import drive
    drive.mount('/content/drive')

Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8gdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com
Enter your authorization code:
.....
Mounted at /content/drive

[3] data = pd.read_csv('/content/drive/My Drive/Colab Notebooks/lab1/glass.csv') #Reading the data from glass.csv

[4] X = data.drop(["Type"], axis=1)
    Y = data["Type"]

[5] Label_Encoder = LabelEncoder()
    data["Type"] = Label_Encoder.fit_transform(data["Type"]) #Label Encoder to convert non numeric to numeric feature

```


In this program we read the glass.csv from the drive and we drop the type column and we use the label encoder to convert the non numeric to numeric feature

```
[5] Label_Encoder = LabelEncoder()
data["Type"] = Label_Encoder.fit_transform(data["Type"]) #label Encoder to convert non numeric to numeric feature

data=data.apply(lambda x: x.fillna(x.mean()),axis=0) #Filling null values with mean using lamda function
print(data.isnull().sum())
```

```
RI      0
Na      0
Mg      0
Al      0
Si      0
K       0
Ca      0
Ba      0
Fe      0
Type    0
dtype: int64
```

Here we print the data

 lab1_5.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[8] X_train, X_test, Y_train, y_test= train_test_split(X, Y, test_size=0.4, random_state=0) #splitting train and test data
```

```
[9] gnb = GaussianNB()

k = gnb.fit(X_train, Y_train).predict(X_test)
gnb.fit(X_train, Y_train)
k = gnb.predict(X_test)
Accuracy = gnb.score(X_test, y_test)
print("Gaussian accuracy is:",Accuracy)
```

```
Gaussian accuracy is: 0.313953488372093
```

```
[11] knn = KNeighborsClassifier(n_neighbors = 3)
knn.fit(X_train, Y_train)
k = knn.predict(X_test)
Accuracy = knn.score(X_test, y_test)
print("Knn accuracy is:",Accuracy)
```

```
Knn accuracy is: 0.6511627906976745
```

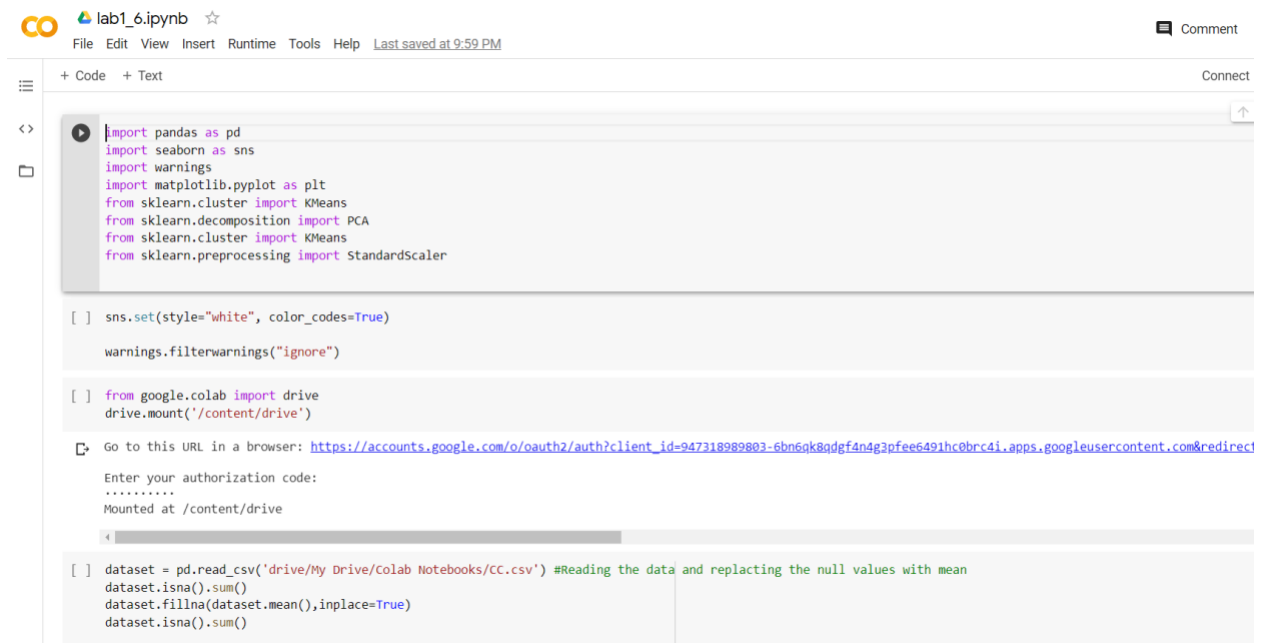
```
svc = SVC()
svc.fit(X_train, Y_train)
k = svc.predict(X_test)
Accuracy = svc.score(X_test, y_test)
print("svm accuracy is:", Accuracy)
```

```
svm accuracy is: 0.38372093023255816
```

Here we split the train data and test data. And we use fit function to fit the data and we use .predict to predict the data instances and print the outputs for naïve bayes, svm and knn algorithms.

So Here Knn algorithm gives the better result than remaining.

6)



```

import pandas as pd
import seaborn as sns
import warnings
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

[ ] sns.set(style="white", color_codes=True)

warnings.filterwarnings("ignore")

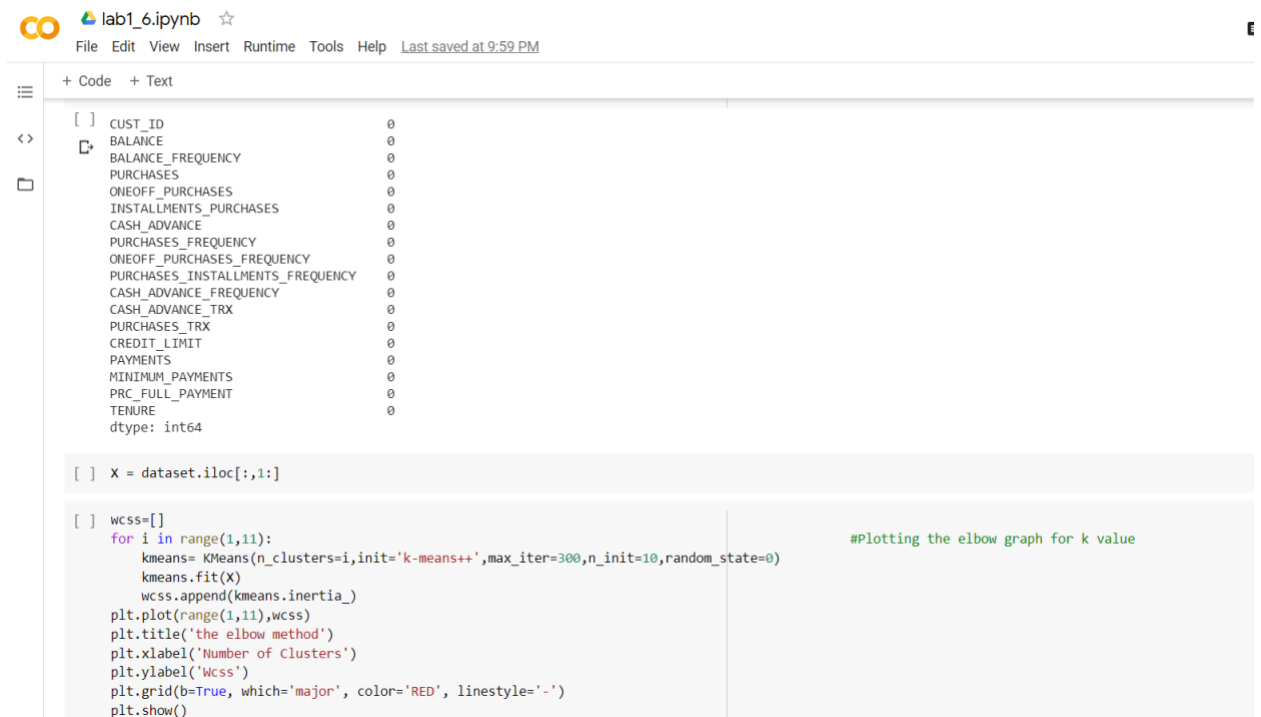
[ ] from google.colab import drive
drive.mount('/content/drive')

Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redirect
Enter your authorization code:
.....
Mounted at /content/drive

[ ] dataset = pd.read_csv('drive/My Drive/colab Notebooks/CC.csv') #Reading the data and replacting the null values with mean
dataset.isna().sum()
dataset.fillna(dataset.mean(),inplace=True)
dataset.isna().sum()

```

In this program we read the cc.csv from the drive and replace the null the values with mean by using .mean().



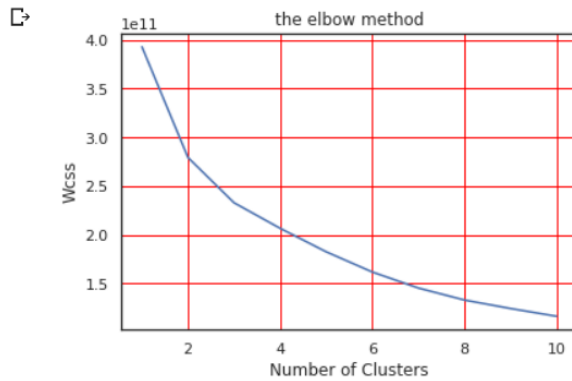
```

[ ] CUST_ID          0
BALANCE             0
BALANCE_FREQUENCY  0
PURCHASES           0
ONEOFF_PURCHASES    0
INSTALLMENTS_PURCHASES  0
CASH_ADVANCE         0
PURCHASES_FREQUENCY  0
ONEOFF_PURCHASES_FREQUENCY  0
PURCHASES_INSTALLMENTS_FREQUENCY  0
CASH_ADVANCE_FREQUENCY  0
CASH_ADVANCE_TRX     0
PURCHASES_TRX        0
CREDIT_LIMIT         0
PAYMENTS             0
MINIMUM_PAYMENTS     0
PRC_FULL_PAYMENT     0
TENURE              0
dtype: int64

[ ] X = dataset.iloc[:,1:]

[ ] wcss=[]
for i in range(1,11):
    kmeans= KMeans(n_clusters=i,init='k-means++',max_iter=300,n_init=10,random_state=0) #Plotting the elbow graph for k value
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
plt.plot(range(1,11),wcss)
plt.title('the elbow method')
plt.xlabel('Number of Clusters')
plt.ylabel('wcss')
plt.grid(b=True, which='major', color='RED', linestyle='-')
plt.show()

```



```
[ ] # K value is 3
    ##building the model
    from sklearn.cluster import KMeans
    nclusters = 3 # this is the k in kmeans
    km = KMeans(n_clusters=nclusters)
    km.fit(X)

    KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
            n_clusters=3, n_init=10, n_jobs=None, precompute_distances='auto',
            random_state=None, tol=0.0001, verbose=0)
```

Here we plot the elbow graph for k value by using the `.plot()`. Once the value of k is known we build the model with the k value.

lab1_6.ipynb ☆
File Edit View Insert Runtime Tools Help Last saved at 9:59 PM

```
+ Code + Text

[ ] # predict the cluster for each data point
    y_cluster_kmeans = km.predict(X)
    from sklearn import metrics
    score = metrics.silhouette_score(X, y_cluster_kmeans)
    print(score)

0.4654251280958053

[ ] #Applying Standardization and PCA
    scaler = StandardScaler()
    scaler.fit(X)
    x_scaler=scaler.transform(X)
    pca = PCA(2)
    x_pca = pca.fit_transform(x_scaler)

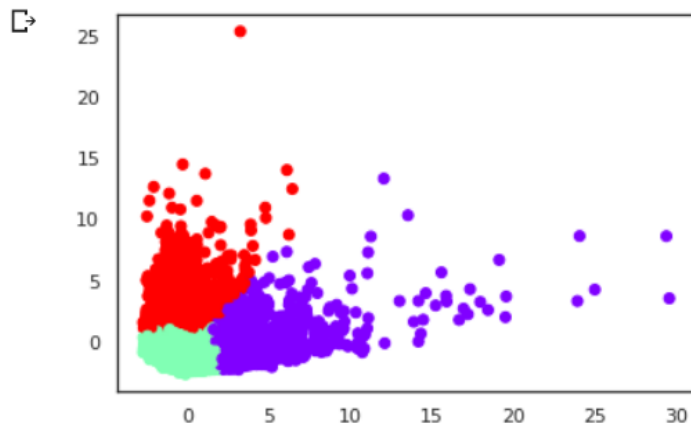
    #From elbow method choose clusters as 3
    km = KMeans(n_clusters=3)
    km.fit(x_pca)
    y_cluster_kmeans= km.predict(x_pca)

    #Calculating silhouette score
    score = metrics.silhouette_score(x_pca, y_cluster_kmeans)

[ ] plt.scatter(x_pca[:,0],x_pca[:,1],c=y_cluster_kmeans,cmap='rainbow')
    plt.show()
```

And we evaluate the silhouette score by using the `silhouette_score()`, and we print the score. After that we apply standardization and PCA

```
[ ] plt.scatter(x_pca[:,0],x_pca[:,1],c=y_cluster_kmeans,cmap='rainbow')
plt.show()
```



7)

lab1_7.ipynb
☆
File Edit View Insert Runtime Tools Help

+ Code + Text

```
[ ] import nltk
from nltk.util import ngrams
from nltk.stem import PorterStemmer
from nltk.stem import WordNetLemmatizer
from nltk import wordpunct_tokenize, pos_tag, ne_chunk
import requests
from bs4 import BeautifulSoup

nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('wordnet')
nltk.download('maxent_ne_chunker')
nltk.download('words')

[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data] Package punkt is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data] /root/nltk_data...
[nltk_data] Package averaged_perceptron_tagger is already up-to-
[nltk_data] date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Package wordnet is already up-to-date!
[nltk_data] Downloading package maxent_ne_chunker to
[nltk_data] /root/nltk_data...
[nltk_data] Package maxent_ne_chunker is already up-to-date!
[nltk_data] Downloading package words to /root/nltk_data...
[nltk_data] Package words is already up-to-date!
True

[ ] from google.colab import drive
drive.mount('/content/drive')

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).

[ ] f = open("/content/drive/My Drive/Colab Notebooks/lab1/nlp_input.txt","r", encoding="cp1252")
text=f.read()
```

In this program we download nltk files and we read the nlp_input.txt file.

```

▶ lemmatizer=WordNetLemmatizer()
  for w in wtokens:
    print(lemmatizer.lemmatize(w))

```

```

↳ and
  lambda
  pair
  and
  we
  can
  see
  which
  pair
  ha
  the
  lowest
  associated
  error
  .
  we
  can
  see

```

We perform lemmatization which is similar to stemming but gives meaningful words

```

▶ wtokens=nltk.word_tokenize(text)
  tokens=nltk.sent_tokenize(text)
  for w in wtokens:
    print(w)

```

```

↳ alpha
  and
  lambda
  pairs
  and
  we
  can
  see
  see
  which
  pair
  has
  the
  lowest
  associated
  error
  .
  We
  can
  see
  that
  the
  R
  mean-squared
  values

```

Here wtokens gives each word as a token from the text.

```

[ ] trigrams=list(ngrams(wtokens,3))
    print(trigrams)

```

```

↳ [('Regression', 'analysis', 'is'), ('analysis', 'is', 'a'), ('is', 'a', 'statistical'), ('a', 'statistical', 'technique'), ('statistical', 'technique', 'that'),

```

Here trigrams gives the consecutive sequence of 3 words from the sentence.

```
[ ] wfrequency = nltk.FreqDist(trigrams)

Topten = wfrequency.most_common(10)

print("Top 10 Trigrams:\n", Topten)
```

Top 10 Trigrams:
 [(('we', 'need', 'to'), 3), (('the', 'coefficients', '.'), 3), (('?', '?', '='), 3), (('to', 'find', 'the'), 2), (('find', 'the', 'optimal'), 2), (('over', 'a', 'number'), 2),

It displays most repeated trigrams with their count.

```
l=[]
for w in tokens:
    for a,b in Topten:
        if a in ngrams(w.split(),3):
            print(a,w)
            l.append(w)
            break
print("concatinated Text")
print(l)
```

['we', 'need', 'to'] First we need to understand the basics of regression and what parameters of the equation are changed when using a specific model.
 ['to', 'find', 'the'] Visualization of the squared error (from Setosa.io)
 The equation for this model is referred to as the cost function and is a way to find the optimal error by minimizing and measuring it.
 ['to', 'find', 'the'] The gradient descent algorithm is used to find the optimal cost function by going over a number of iterations.
 ['we', 'need', 'to'] But the data we need to define and analyze is not always so easy to characterize with the base OLS model.
 ['to', 'each', 'other'] Equation for least ordinary squares
 One situation is the data showing multi-collinearity, this is when predictor variables are correlated to each other and to the response variable.
 ['penalty', 'term', 'to'] To produce a more accurate model of complex data we can add a penalty term to the OLS equation.
 ['penalty', 'term', 'to'] These are known as L1 regularization(lasso regression) and L2 regularization(ridge regression).The best model we can hope to come up with minimizes both th
 Ridge regression uses L2 regularization which adds the following penalty term to the OLS equation.
 ['we', 'need', 'to'] = 1 denotes lasso)
 Performing Elastic Net regression
 Performing Elastic Net requires us to tune parameters to identify the best alpha and lambda values and for this we need to use the caret package.
 ('over', 'a', 'number') We will tune the model by iterating over a number of alpha and lambda pairs and we can see which pair has the lowest associated error.
 concatinated Text
 ['First we need to understand the basics of regression and what parameters of the equation are changed when using a specific model.', 'Visualization of the squared error (from Setos

Here the above text file is taken and finding the repeated trigrams and append is used to concatenate. Finally we display the concatenated results.

8)

lab1_8.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[1] from pathlib import Path
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn import linear_model
from sklearn.metrics import mean_squared_error
```

```
[2] from google.colab import drive
drive.mount('/content/drive')
```

Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6gk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redirect_uri=https://colab.research.google.com/&response_type=code
 Enter your authorization code:

 Mounted at /content/drive

```
[3] data = pd.read_csv(Path('/content/drive/My Drive/Colab Notebooks/lab1/winequality-red.csv'))
```

In this program we read the winequality-red.csv from the drive.

```
lab1_8.ipynb ☆
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[4] y = np.log(data.quality)
    X = data.drop(['quality'], axis=1)

    X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42, test_size=.33)

    lr = linear_model.LinearRegression()
    model = lr.fit(X_train, y_train)

    # Evaluate the performance and visualize results
    print("R2 is: ", model.score(X_test, y_test))
    predictions = model.predict(X_test)

    print('Rmse is: ', mean_squared_error(y_test, predictions))

R2 is:  0.3433962675485108
Rmse is:  0.013811674843193872

[5] # Null values
    null = pd.DataFrame(data.isnull().sum().sort_values(ascending=False))
    null.columns = ['Null Count']
    null.index.name = 'Feature'
    print(null)

    # handling missing value
    data = data.select_dtypes(include=[np.number]).interpolate().dropna()
    print(sum(data.isnull().sum() != 0))
```

we drop the quality field. And we split the train data and test data. We use .score and .predict to evaluate the performance and visualize the results. Isnull() is used for null values.

```
lab1_8.ipynb ☆
File Edit View Insert Runtime Tools Help All changes saved Comment
RAM
Disk

+ Code + Text

[5] Feature          Null Count
    quality          0
    alcohol          0
    sulphates        0
    pH               0
    density          0
    total sulfur dioxide 0
    free sulfur dioxide 0
    chlorides        0
    residual sugar   0
    citric acid       0
    volatile acidity 0
    fixed acidity     0
    0

y = np.log(data.quality)
X = data.drop(['quality'], axis=1)

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42, test_size=.33)

lr = linear_model.LinearRegression()
model = lr.fit(X_train, y_train)

# Evaluate the performance and visualize results
print("R2 is: ", model.score(X_test, y_test))
predictions = model.predict(X_test)

print('Rmse is: ', mean_squared_error(y_test, predictions))

R2 is:  0.3433962675485108
Rmse is:  0.013811674843193872
```

In this field we evaluate the model using RMSE and R2 and the results are same

Conclusion

We have learned several methods and approaches while implementing lab programs such as Classification, Clustering, Web scraping, Data Visualization, application of Natural Language Processing and Multiple Regression.