PYTHON LAB ASSIGNMENT 1

BY

Team 2

Akhil Teja Kanugolu – 11

Geetanjali Makineni – 13

Github Link: https://github.com/akhilkanugolu/Lab1

Video Link: https://umkc.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=d7b583f5-23b9-4d54-a4c0-ab7b0018499f

Wiki Link: https://github.com/akhilkanugolu/Lab1/wiki

Introduction:

In this Lab we use the List, dictionaries and Over ridding functions, Multiple Inheritance in python and built the models from the dataset using scikit-learn library for the Inbuilt packages . We got the Data set from "Super Data Science", Later preprocessed the Data we obtained next applied feature scaling to the data. Finally analyzed the data using different models Regression, Clustering and Classification and visualized the data.

Objective:

- Lists, Dictionaries, Loops, Classes, Method Overriding, Multiple Inheritance, Beautiful Soup Package for Web scrapping.
- And Picking up of Dataset, Understanding the Features or Attributes.
- Classification Algorithms like Naïve Bayes, SVM, KNN.
- Regression Algorithm Multiple Linear Regression- Handling Nulls and Selecting Independent Variables (Predictors) using correlation to predict target Variable (Independent Variable). And encoding the categorical variables. Used R2 and RMSE to predict efficiency.
- Clustering K-Means Algorithm and Used elbow method to get the clusters, silhouette score to explain the model efficiency.
- Get know about Tokenize, Lemmatization, Trigrams.

Requirements:

- Anaconda Interpreter
- Spyder
- Python 3.5

Approaches and Methods:

1. Used Lists, Dictionaries for storing data. Loops are used to append the data into the list.
2. Classes, Method Overriding used for the Library Management system and Multiple Inheritance used for getting data of super class
3. Beautiful Soup Package for Web scrapping of URL.
4. Exploratory Data Analysis was used to preprocess the Data like handling null values with mean values or median or by dropped the null values.
5. ML techniques like regression, classification and clustering we use the scikit-learn library.
6. For Calculating the RMSE, Elbow Method variance and Silhoutte scores we used the predefined methods in the scikit-learn library.

Workflow:

Task 1:

Editor - C:\Users\geeta\OneDrive\Desktop\python lab\lab1_q1.py

lab1_q1.py | lab1_q2.py | lab1_q3.py | lab1_q4.py | lab1_q5.py | lab1_q6.py | lab1_q7.py

```python
"""1)Given a collection of integers that might contain
duplicates, nums, return all possible subsets. Do not include null subset"""

#Calling Sublist function
def sublist(inp_list):
    list1=[]
    n=len(inp_list)
    for i in range(2**n):
        subset=[]
        for j in range(n):
            if (i&(1<<j))!=0:
                subset.append(inp_list[j])
        if subset not in list1:
            list1.append(subset)
    return list1[1:]

#Give Input List
input_list=[1,2,2]
print("Output\n",sublist(input_list))
```

Output:

IPython console

Console 1/A

```
   ...: def sublist(inp_list):
   ...:     list1=[]
   ...:     n=len(inp_list)
   ...:     for i in range(2**n):
   ...:         subset=[]
   ...:         for j in range(n):
   ...:             if (i&(1<<j))!=0:
   ...:                 subset.append(inp_list[j])
   ...:         if subset not in list1:
   ...:             list1.append(subset)
   ...:     return list1[1:]
   ...:
   ...:
   ...: #Give Input List
   ...: input_list=[1,2,2]
   ...: print("Output\n",sublist(input_list))
Output
 [[1], [2], [1, 2], [2, 2], [1, 2, 2]]
```

Task 2:

Editor - C:\Users\geeta\OneDrive\Desktop\python lab\lab1_q2.py

lab1_q1.py | lab1_q2.py | lab1_q3.py | lab1_q4.py | lab1_q5.py | lab1_q6.py | lab

```python
def concate(dict1, dict2):
    result_dict = {**dict1, **dict2}
    return result_dict

# Driver code
dict1 = {'p': 8, 'r': 5}
dict2 = {'q': 2, 's': 9}
dict3 = concate(dict1, dict2)
print(dict3)

for key, value in sorted(dict3.items(), key=lambda item: item[1]):
    print("%s: %s" % (key, value))

```

Output:

IPython console

Console 1/A

```
In [6]: def concate(dict1, dict2):
   ...:     result_dict = {**dict1, **dict2}
   ...:     return result_dict
   ...:
   ...:
   ...: # Driver code
   ...: dict1 = {'p': 8, 'r': 5}
   ...: dict2 = {'q': 2, 's': 9}
   ...: dict3 = concate(dict1, dict2)
   ...: print(dict3)
   ...:
   ...: for key, value in sorted(dict3.items(), key=lambda item: item[1]):
   ...:     print("%s: %s" % (key, value))
{'p': 8, 'r': 5, 'q': 2, 's': 9}
q: 2
r: 5
p: 8
s: 9
```

Task 3:

Editor - C:\Users\geeta\OneDrive\Desktop\python lab\lab1_q3.py

lab1_q1.py lab1_q2.py lab1_q3.py lab1_q4.py lab1_q5.py lab1_q6.py lab1_q7.py lab1_q8.py

```python

# Person Class
class Person():
#defined constructor
    def __init__(self,full_name,emailadd,p_id,phn_num):
        self.Pname = full_name
        self.email = emailadd
        self.p_id = p_id
        self.pnum = phn_num
#prints name of a person
    def getfname(self):
        print("Name : ", self.Pname)
#prints phone number
    def getphnum(self):
        print("Phone number : ", self.pnum)
#prints email address
    def getemail(self):
        print("Person email address : ", self.email)
# defined private member
    def __get_id(self):
        print("Person ID :", self.p_id)

class Department():
# Defined __init__ constructor
    def __init__(self,dept,year):
        self.dept=dept
        self.Byear=year
#prints Department
    def getdept(self):
        print("Department :",self.dept)
#prints Batch Year
    def getyear(self):
        print("Batch Year :",self.Byear)

```

Editor - C:\Users\geeta\OneDrive\Desktop\python lab\lab1_q3.py

lab1_q1.py lab1_q2.py lab1_q3.py lab1_q4.py lab1_q5.py lab1_q6.py lab1_q7.py lab1_q8.py

```python
        print("Batch Year :",self.Byear)

# Defined Book Details
class Book():
# Defined __init__ constructor
    def __init__(self,Book_name):
        from datetime import date,timedelta
        Booklist=["Book1","Book2","Book3"]
        BookAuthlist=["Auth1","Auth2","Auth3"]
        BookGenre=["Genre1","Genre2","Genre3"]
        self.Bname = Book_name
        if self.Bname in Booklist:
            X=input("Book is Available--Do you want to proceed Yes-Y or No-N:").upper()
            if X==('Y'):
                self.idate = date.today()
                self.rdate = date.today()+timedelta(days=15)
                self.Bauth = BookAuthlist[Booklist.index(self.Bname)]
                self.Bgenre = BookGenre[Booklist.index(self.Bname)]
                self.getbookname()
                self.getissueddate()
                self.getreturndate()
            else:
                print("Your Booking is Cancelled")

        else:
            print("Book Not Available")
    #print book name
    def getbookname(self):
        print("Book Name :", self.Bname)
    #print author of the book
    def getbookauth(self):
        print("Author of the book :", self.Bauth)
    #print Book Genre
    def getbookgenre(self):
        print("Book Genre :", self.Bgenre)
    # print Book issued date
    def getissueddate(self):
```

Editor - C:\Users\geeta\OneDrive\Desktop\python lab\lab1_q3.py

lab1_q1.py | lab1_q2.py | lab1_q3.py | lab1_q4.py | lab1_q5.py | lab1_q6.py | lab1_q7.py | lab1_q8.py

```python
    # print Book returned date
    def getreturndate(self):
        print("Book should be returned on :",self.rdate)

# Multiple inheritance, Faculty Class inherits Person,Department,Book

class Faculty(Person,Department,Book):
    def __init__(self,full_name,emailadd,p_id,phn_num,dept,year,Book_name):
        self.getprofession()
        super().__init__(full_name,emailadd,p_id,phn_num)
        Department.__init__(self,dept,year)
        Book.__init__(self,Book_name)

    def getprofession(self):
        print("Welcome UMKC Faculty")

# Multiple inheritance, Student Class inherits Person,Department,Book
class Student(Person,Department,Book):
    def __init__(self,full_name,emailadd,p_id,phn_num,dept,year,Book_name):
        self.getprofession()
        super().__init__(full_name,emailadd,p_id,phn_num)
        Department.__init__(self,dept,year)
        Book.__init__(self,Book_name)

    #Method OverRidding
    def getprofession(self):
        print("Welcome UMKC Student")

stud1 = Student("Akhil Teja Kanugolu","akhil@gmail.com","16297766","9842456635","ECE","2014-18","Book2")
stud1.getprofession()
print("################# Student Details  ###########################")
stud1.getfname()
stud1.getemail()
stud1.getphnum()
print("################# Department Details  #######################")
stud1.getdept()
```

Editor - C:\Users\geeta\OneDrive\Desktop\python lab\lab1_q3.py

lab1_q1.py | lab1_q2.py | lab1_q3.py | lab1_q4.py | lab1_q5.py | lab1_q6.py | lab1_q7.py | lab1_q8.py

```python

stud1 = Student("Akhil Teja Kanugolu","akhil@gmail.com","16297766","9842456635","ECE","2014-18","Book2")
stud1.getprofession()
print("################# Student Details  ###########################")
stud1.getfname()
stud1.getemail()
stud1.getphnum()
print("################# Department Details  #######################")
stud1.getdept()
stud1.getyear()
print("################# Book Details  #############################")
stud1.getbookname()
stud1.getbookauth()
stud1.getbookgenre()
print("################ Book Issuing Details : #####################")
stud1.getissueddate()
stud1.getreturndate()


fac1 = Faculty("Geetanjali Makineni","geeta@gmail.com","16290659","8164420251","CSE","2001-18","Book1")
fac1.getprofession()
print("################# Faculty Details  ###########################")
fac1.getfname()
fac1.getemail()
fac1.getphnum()
print("################# Department Details  #######################")
stud1.getdept()
stud1.getyear()
print("################# Book Details  #############################")
fac1.getbookname()
fac1.getbookauth()
fac1.getbookgenre()
print("################ Book Issuing Details : #####################")
fac1.getissueddate()
fac1.getreturndate()


```

Output:

If the Student tried to check for the book in the library data base→If book is available, he can procced with “Y”

```
Welcome UMKC Student

Book is Available--Do you want to proceed Yes-Y or No-N:Y
Book Name : Book2
Book issued on : 2020-03-10
Book should be returned on : 2020-03-25
Welcome UMKC Student
################# Student Details  ###########################
Name :  Akhil Teja Kanugolu
Person email address :  akhil@gmail.com
Phone number :  9842456635
################# Department Details  #######################
Department : ECE
Batch Year : 2014-18
################# Book Details  #############################
Book Name : Book2
Author of the book : Auth2
Book Genre : Genre2
################ Book Issuing Details : #####################
Book issued on : 2020-03-10
Book should be returned on : 2020-03-25
```

If the Student tried to check for the book in the library data base→If book is available, But he don’t want to proceed “N”

```
Welcome UMKC Student

Book is Available--Do you want to proceed Yes-Y or No-N:n
Your Booking is Cancelled
```

If book Not Available, then we will get the following screen.

```
In [12]: stud1 = Student("Akhil Teja
Kanugolu","akhil@gmail.com","16297766","9842456635","ECE","2014-18","Book5")
Welcome UMKC Student
Book Not Available
```

If the Faculty tried to check for the book in the library data base→If book is available, he can procced with “Y”

```
    .... fac1.getreturndate()
Welcome UMKC Faculty

Book is Available--Do you want to proceed Yes-Y or No-N:Y
Book Name : Book1
Book issued on : 2020-03-10
Book should be returned on : 2020-03-25
Welcome UMKC Faculty
################# Faculty Details  ###########################
Name :  Geetanjali Makineni
Person email address :  geeta@gmail.com
Phone number :  8164420251
################# Department Details  #######################
Department : ECE
Batch Year : 2014-18
################# Book Details  #############################
Book Name : Book1
Author of the book : Auth1
Book Genre : Genre1
################ Book Issuing Details : #####################
Book issued on : 2020-03-10
Book should be returned on : 2020-03-25
```

If the Faculty tried to check for the book in the library data base→If book is available, but he don’t want to proceed “N”

```
In [11]: fac1 = Faculty("Geetanjali
Makineni","geeta@gmail.com","16290659","8164420251","CSE","2001-18","Book1")
Welcome UMKC Faculty

Book is Available--Do you want to proceed Yes-Y or No-N:N
Your Booking is Cancelled
```

Task 4:

Editor - C:\Users\geeta\OneDrive\Desktop\python lab\lab1_q4.py

lab1_q1.py lab1_q2.py lab1_q3.py* lab1_q4.py lab1_q5.py lab1_q6.py lab1_q7.py lab1_q8.p

```python


from bs4 import BeautifulSoup
import urllib.request


url = "https://catalog.umkc.edu/course-offerings/graduate/comp-sci/"
source_code = urllib.request.urlopen(url)
soup = BeautifulSoup(source_code, "html.parser")


for block in soup.find_all('div',{"class":"courseblock"}):
    title = block.find('span',{'class':'title'})
    over_view = block.find('p',{'class':'courseblockdesc'})
    result = "Course Title:"+str(title.text) + "\n" +"Course Overview:"+ str(over_view.text.strip())
    print(result+"\n")

```

Output:

Web Scrapping using Bs4 to pull course title and description.

IPython console

Console 1/A

```
   ...:     over_view = block.find('p',{'class':'courseblockdesc'})
   ...:     result = "Course Title:"+str(title.text) + "\n" +"Course Overview:"+ str(over_view.text.strip())
   ...:     print(result+"\n")
Course Title:Discrete Structures Review for Graduate Students
Course Overview:A review of mathematical logic, sets, relations, functions, mathematical induction, and
algebraic structures with emphasis on computing applications. Recurrence relations and their use in the analysis
of algorithms. Graphs, trees, and network flow models. Introduction to Finite state machines, grammars, and
automata. Students must have completed College Algebra before taking this course.

Course Title:Operating Systems Review for Graduate Students
Course Overview:This course covers concurrency and control of asynchronous processes, deadlocks, memory
management, processor and disk scheduling, parallel processing, and file system organization in operating
systems.

Course Title:Advanced Data Structures and Analysis of Algorithms Review for Graduate Students
Course Overview:A review of linear and hierarchical data structures, including stacks, queues, lists, trees,
priority queues, advanced tree structures, hashing tables, dictionaries and disjoint-sets. Asymptotic analysis
techniques and algorithms: from design strategy (such as greedy, divide-and-conquer, and dynamic programming) to
problem areas (such as searching, sorting, shortest path, spanning trees, transitive closures, graph algorithms,
and string algorithms) arriving at classical algorithms with efficient implementation. Introduction to the basic
concepts of complexity theory and NP-complete theory. Students must have taken courses in Linear Algebra,
Discrete Structures, Data Structures, and Applied Probability before taking this course.

Course Title:Optical Fiber Communications
Course Overview: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, modulation and
transmission of information over optical fibers, design of optical networks, single and multihop fiber LANs,
optical carrier systems.
```

Task 5:

Editor - C:\Users\geeta\OneDrive\Desktop\python lab\lab1_q5.py

lab1_q1.py lab1_q2.py lab1_q3.py* lab1_q4.py lab1_q5.py lab1_q6.py lab1_q7.py lab1_q8.py

```python

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# visualization
import seaborn as sns

# Importing the dataset
social_data = pd.read_csv('Social_Network_Ads.csv')

#describing data
social_data.describe()

#Dropping User ID
social_data = social_data.drop(['User ID'], axis=1)

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
social_data_train, social_data_test = train_test_split(social_data, test_size = 0.25, random_state = 0)
combine = [social_data_train, social_data_test]

####Correlating numerical features
g = sns.FacetGrid(social_data_train, col='Purchased')
g.map(plt.hist, 'Age', bins=20)

g = sns.FacetGrid(social_data_train, col='Purchased')
g.map(plt.hist, 'EstimatedSalary', bins=20)

# Encoding categorical data
for dataset in combine:
    dataset['Gender'] = dataset['Gender'].map( {'Female': 1, 'Male': 0} ).astype(int)

X_train=social_data_train.drop('Purchased',axis=1)
Y_train=social_data_train['Purchased']
```

Editor - C:\Users\geeta\OneDrive\Desktop\python lab\lab1_q5.py

lab1_q1.py lab1_q2.py lab1_q3.py* lab1_q4.py lab1_q5.py lab1_q6.py lab1_q7.py lab1_q8.py

```python
X_train=social_data_train.drop('Purchased',axis=1)
Y_train=social_data_train['Purchased']
X_test=social_data_test.drop('Purchased',axis=1)
Y_test=social_data_test['Purchased']

# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

#Fitting Naive Bayes
from sklearn.naive_bayes import GaussianNB
nb=GaussianNB()
nb.fit(X_train, Y_train)
nb_pred= nb.predict(X_test)
acc_nb= round(nb.score(X_train, Y_train) * 100, 2)
print("Naive Bayes accuracy is:", acc_nb)

# Fitting SVM to the Training set
from sklearn.svm import SVC
svc = SVC()
svc.fit(X_train, Y_train)
svc_pred= svc.predict(X_test)
acc_svc= round(svc.score(X_train, Y_train) * 100, 2)
print("svm accuracy is:", acc_svc)

# Fitting KNN to the Training set
from sklearn.neighbors import KNeighborsClassifier
knn= KNeighborsClassifier(n_neighbors= 3)
knn.fit(X_train, Y_train)
knn_pred= knn.predict(X_test)
acc_knn= round(knn.score(X_train, Y_train) * 100, 2)
print("KNN accuracy is:",acc_knn)

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
```

```python

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cmNb = confusion_matrix(Y_test, nb_pred)
cmSVC = confusion_matrix(Y_test, svc_pred)
cmKnn = confusion_matrix(Y_test, knn_pred)

print("Naive Bayes\n",cmNb)
print("SVC\n",cmSVC)
print("Knn\n",cmKnn)
```

Output:

Social Data set

Index	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0
5	15728773	Male	27	58000	0
6	15598044	Female	27	84000	0
7	15694829	Female	32	150000	1
8	15600575	Male	25	33000	0
9	15727311	Female	35	65000	0
10	15570769	Female	26	80000	0
11	15606274	Female	26	52000	0
12	15746139	Male	20	86000	0
13	15704987	Male	32	18000	0
14	15628972	Male	18	82000	0
15	15697686	Male	29	80000	0
16	15733883	Male	47	25000	1
17	15617482	Male	45	26000	1
18	15704583	Male	46	28000	1
19	15621083	Female	48	29000	1
20	15649487	Male	45	22000	1

Social Data set After Dropping the user ID

Index	Gender	Age	EstimatedSalary	Purchased
0	Male	19	19000	0
1	Male	35	20000	0
2	Female	26	43000	0
3	Female	27	57000	0
4	Male	19	76000	0
5	Male	27	58000	0
6	Female	27	84000	0
7	Female	32	150000	1
8	Male	25	33000	0
9	Female	35	65000	0
10	Female	26	80000	0
11	Female	26	52000	0
12	Male	20	86000	0
13	Male	32	18000	0
14	Male	18	82000	0
15	Male	29	80000	0
16	Male	47	25000	1
17	Male	45	26000	1
18	Male	46	28000	1
19	Female	48	29000	1
20	Male	45	22000	1

Social Data set after converting the categorical variable gender into numerical

Female-1 and Male-0

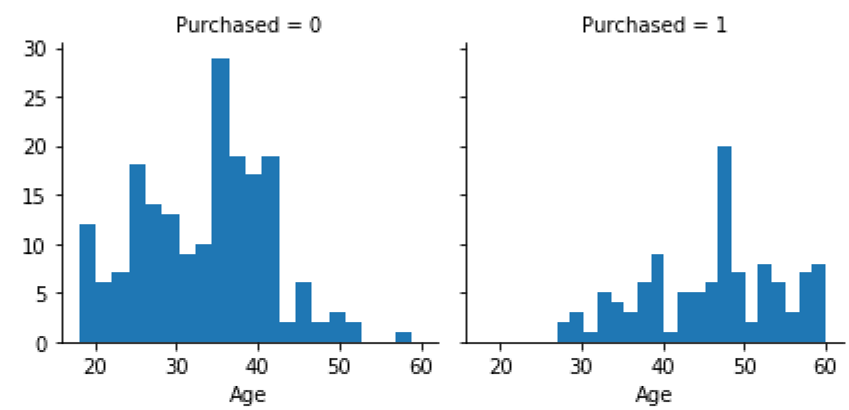
Index	Gender	Age	EstimatedSalary	Purchased
132	0	30	87000	0
309	1	38	50000	0
341	0	35	75000	0
196	1	30	79000	0
246	1	35	50000	0
60	0	27	20000	0
155	1	31	15000	0
261	0	36	144000	1
141	1	18	68000	0
214	0	47	43000	0
37	0	30	49000	0
134	1	28	55000	0
113	0	37	55000	0
348	0	39	77000	0
12	0	20	86000	0
59	1	32	117000	0
293	0	37	77000	0
140	0	19	85000	0
206	1	55	130000	1
199	0	35	22000	0
176	1	35	47000	0

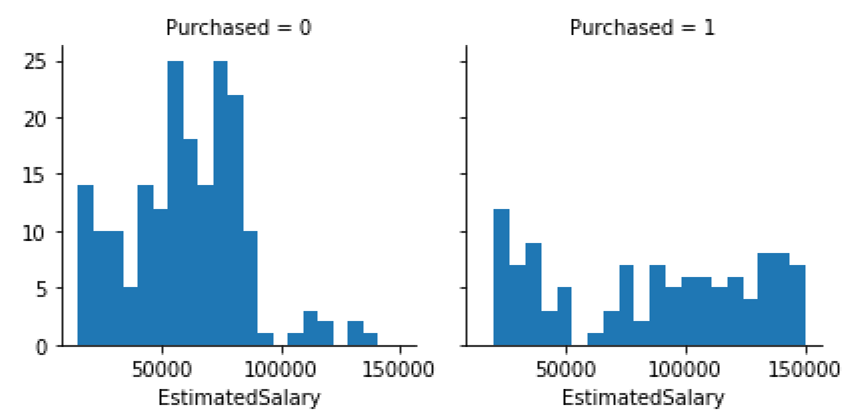
Got the Features of the columns of social data set

```
In [17]: social_data.describe()
Out[17]:
```

	User ID	Age	EstimatedSalary	Purchased
count	4.000000e+02	400.000000	400.000000	400.000000
mean	1.569154e+07	37.655000	69742.500000	0.357500
std	7.165832e+04	10.482877	34096.960282	0.479864
min	1.556669e+07	18.000000	15000.000000	0.000000
25%	1.562676e+07	29.750000	43000.000000	0.000000
50%	1.569434e+07	37.000000	70000.000000	0.000000
75%	1.575036e+07	46.000000	88000.000000	1.000000
max	1.581524e+07	60.000000	150000.000000	1.000000

Plot the histogram for age and Estimated Salary with Purchased





```
Naive Bayes accuracy is: 89.0
svm accuracy is: 90.33
KNN accuracy is: 91.67
Naive Bayes
 [[66  2]
 [ 7 25]]
SVC
 [[64  4]
 [ 3 29]]
Knn
 [[64  4]
 [ 4 28]]
```

By observing the accuracy, confusion matrix of social Dataset using Naïve Bayes, SVM, KNN we can say that KNN model with 91.67 accuracy is fit for this dataset.

Task 6:

Editor - C:\Users\geeta\OneDrive\Desktop\python lab\lab1_q6.py

lab1_q1.py | lab1_q2.py | lab1_q3.py* | lab1_q4.py | lab1_q5.py | lab1_q6.py | lab1_q7.py | lab1_q8.py

```python
# K-Means Clustering

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import pandas as pd

# Importing the dataset
dataset = pd.read_csv('Mall_Customers.csv')
dataset.isna().sum()
dataset.fillna(dataset.mean(),inplace=True)
dataset.isna().sum()

X = dataset.iloc[:, [3, 4]].values
y = dataset.iloc[:, 3].values

# Using the elbow method to find the optimal number of clusters
from sklearn.cluster import KMeans
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
    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.show()


from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
# Fit on training set only.
scaler.fit(X)
# Apply transform to both the training set and the test set.
```

Editor - C:\Users\geeta\OneDrive\Desktop\python lab\lab1_q6.py

lab1_q1.py | lab1_q2.py | lab1_q3.py* | lab1_q4.py | lab1_q5.py | lab1_q6.py | lab1_q7.py | lab1_q8.py

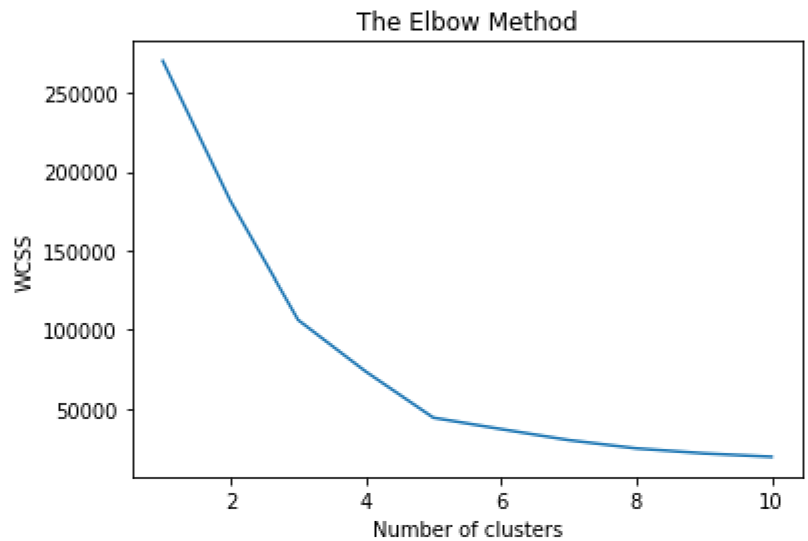
```python
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()


from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
# Fit on training set only.
scaler.fit(X)
# Apply transform to both the training set and the test set.
X= scaler.transform(X)


# Fitting K-Means to the dataset
kmeans = KMeans(n_clusters = 5, init = 'k-means++', random_state = 42)
y_kmeans = kmeans.fit_predict(X)

# predict the cluster for each data point
from sklearn import metrics
score = metrics.silhouette_score(X, y_kmeans)
print('Silhouette score for 5 clusters after scaled',score)

# Visualising the clusters
plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], s = 100, c = 'red', label = 'Cluster 1')
plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], s = 100, c = 'blue', label = 'Cluster 2')
plt.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], s = 100, c = 'green', label = 'Cluster 3')
plt.scatter(X[y_kmeans == 3, 0], X[y_kmeans == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4')
plt.scatter(X[y_kmeans == 4, 0], X[y_kmeans == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 300, c = 'yellow', label = 'Centroids')
plt.title('Clusters of customers')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
```

Output:

Checking for the Nulls for the features

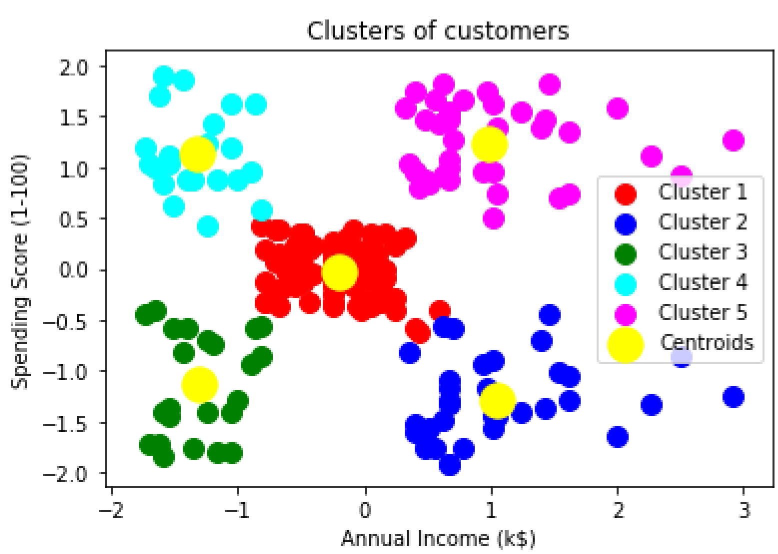
```
Out[22]:
CustomerID                0
Genre                     0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64
```

Used Elbow method to get the number of clusters. From this plot we got the elbow at “5”




```
Silhouette score for 5 clusters after scaled 0.5546571631111091
```

Plot using 5 “Clusters” for K-Mean



Task 7:

Editor - C:\Users\geeta\OneDrive\Desktop\python lab\lab1_q7.py

lab1_q1.py lab1_q2.py lab1_q3.py* lab1_q4.py lab1_q5.py lab1_q6.py lab1_q7.py lab1_q8.py

```python
from bs4 import BeautifulSoup
import urllib.request
import requests
import pandas as pd

search = input('type "s" to start wikiScrap, type "q" to exit:')
if search == 'q' or search == 'Q':
    print("Quiting...")
    exit()
else:
    print("Creating .txt file ...")
    file = open('input.txt', 'a+', encoding='utf-8')
    url ="https://public.boxcloud.com/api/2.0/internal_files/413465789782/versions/437018393782/representations/
    headers= {'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_1) AppleWebKit/537.36 (KHTML, like Gec
    r = requests.get(url,headers=headers)
    soup = BeautifulSoup(r.content, 'html.parser')
    file.write(str(soup))
    print(soup)

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

#Opening the saved input Files
sentence = open('input.txt', encoding="utf8").read()

# Tokenization
stokens = nltk.sent_tokenize(sentence)
wtokens = nltk.word_tokenize(sentence)

print("\nWord  Tokenization:\n")
print(wtokens)
print("\nSentence  Tokenization:\n")
```

Editor - C:\Users\geeta\OneDrive\Desktop\python lab\lab1_q7.py

lab1_q1.py lab1_q2.py lab1_q3.py* lab1_q4.py lab1_q5.py lab1_q6.py lab1_q7.py* lab1_q8.py

```python
print(stokens)

# Lemmatization
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
print("\nPOS / Lemmatization\n")

for t in wtokens:
        print("Lemmatizer:", lemmatizer.lemmatize(t), ",    With POS=a:", lemmatizer.lemmatize(t, pos="a"))

# Trigram
from nltk.util import ngrams
print("\nTrigram\n")
trigrams=ngrams(wtokens,3)
trigram_list=[]
for trigram in trigrams:
    trigram_list.append(trigram)
    print(trigram)

#Caluclating Frequency
wordFreq=nltk.FreqDist(trigram_list)
top_ten=wordFreq.most_common(10)
print("Top 10 Repeated Trigrams\n",top_ten)


#Check sentences
concate=""

for j in range(len(top_ten)):
    for sen in stokens:
        token = nltk.word_tokenize(sen)
        trigrams = list(ngrams(token, 3))
        if top_ten[j][0] in trigrams:
            print("-->",sen)
            concate=concate+" "+sen

print("\n#################### Concatation of Sentences #####################\n",concate)
```

Output:

Used web scraping to get the data from URL and saved the text into input.txt file

```
type "s" to start wikiScrap, type "q" to exit:S
Creating .txt file ...
```

```
Word  Tokenization:

['Regression', 'analysis', 'is', 'a', 'statistical', 'technique', 'that', 'models', 'and', 'approximates',
'the', 'relationship', 'between', 'a', 'dependent', 'and', 'one', 'or', 'more', 'independent', 'variables',
'.', 'This', 'article', 'will', 'quickly', 'introduce', 'three', 'commonly', 'used', 'regression', 'models',
'using', 'R', 'and', 'the', 'Boston', 'housing', 'data-set', ':', 'Ridge', ',', 'Lasso', ',', 'and',
'Elastic', 'Net', '.', 'First', 'we', 'need', 'to', 'understand', 'the', 'basics', 'of', 'regression',
'and', 'what', 'parameters', 'of', 'the', 'equation', 'are', 'changed', 'when', 'using', 'a', 'specific',
'model', '.', 'Simple', 'linear', 'regression', ',', 'also', 'known', 'as', 'ordinary', 'least', 'squares',
'(', 'OLS', ')', 'attempts', 'to', 'minimize', 'the', 'sum', 'of', 'error', 'squared', '.', 'The', 'error',
'in', 'this', 'case', 'is', 'the', 'difference', 'between', 'the', 'actual', 'data', 'point', 'and', 'its',
'predicted', 'value', '.', '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',
'.', 'The', 'gradient', 'descent', 'algorithm', 'is', 'used', 'to', 'find', 'the', 'optimal', 'cost',
'function', 'by', 'going', 'over', 'a', 'number', 'of', 'iterations', '.', 'But', 'the', 'data', 'we',
'need', 'to', 'define', 'and', 'analyze', 'is', 'not', 'always', 'so', 'easy', 'to', 'characterize', 'with',
'the', 'base', 'OLS', 'model', '.', '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', '.', 'To',
'picture', 'this', 'let', '’', 's', 'say', 'we', '’', 're', 'doing', 'a', 'study', 'that', 'looks', 'at',
'a', 'response', 'variable', '?', '—', '?', 'patient', 'weight', ',', 'and', 'our', 'predictor',
'variables', 'would', 'be', 'height', ',', 'sex', ',', 'and', 'diet', '.', 'The', 'problem', 'here', 'is',
'that', 'height', 'and', 'sex', 'are', 'also', 'correlated', 'and', 'can', 'inflate', 'the', 'standard',
'error', 'of', 'their', 'coefficients', 'which', 'may', 'make', 'them', 'seem', 'statistically',
```

```
Sentence  Tokenization:

['Regression analysis is a statistical technique that models and approximates the relationship between a
dependent and one or more independent variables.', 'This article will quickly introduce three commonly used
regression models using R and the Boston housing data-set: Ridge, Lasso, and Elastic Net.', 'First we need
to understand the basics of regression and what parameters of the equation are changed when using a specific
model.', 'Simple linear regression, also known as ordinary least squares (OLS) attempts to minimize the sum
of error squared.', 'The error in this case is the difference between the actual data point and its
predicted value.', 'Visualization of the squared error (from Setosa.io)\nThe 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.',
'The gradient descent algorithm is used to find the optimal cost function by going over a number of
iterations.', 'But the data we need to define and analyze is not always so easy to characterize with the
base OLS model.', 'Equation for least ordinary squares\nOne situation is the data showing multi-
collinearity, this is when predictor variables are correlated to each other and to the response variable.',
'To picture this let’s say we’re doing a study that looks at a response variable?—?patient weight, and our
predictor variables would be height, sex, and diet.', 'The problem here is that height and sex are also
correlated and can inflate the standard error of their coefficients which may make them seem statistically
insignificant.', 'To produce a more accurate model of complex data we can add a penalty term to the OLS
equation.', 'A penalty adds a bias towards certain values.', '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 the bias and the variance:\nRidge regression uses L2 regularization which adds the following penalty
term to the OLS equation.', 'L2 regularization penalty term\nThe L2 term is equal to the square of the
magnitude of the coefficients.', 'In this case if lambda(?)', 'is zero then the equation is the basic OLS
but if it is greater than zero then we add a constraint to the coefficients.', 'This constraint results in
minimized coefficients (aka shrinkage) that trend towards zero the larger the value of lambda.', 'Shrinking
the coefficients leads to a lower variance and in turn a lower error value.', 'Therefore Ridge regression
```

```
Lemmatizer: but ,    With POS=a: but
Lemmatizer: doe ,    With POS=a: does
Lemmatizer: not ,    With POS=a: not
Lemmatizer: reduce ,    With POS=a: reduce
Lemmatizer: the ,    With POS=a: the
Lemmatizer: number ,    With POS=a: number
Lemmatizer: of ,    With POS=a: of
Lemmatizer: variable ,    With POS=a: variables
Lemmatizer: , ,    With POS=a: ,
Lemmatizer: it ,    With POS=a: it
Lemmatizer: rather ,    With POS=a: rather
Lemmatizer: just ,    With POS=a: just
Lemmatizer: shrink ,    With POS=a: shrinks
Lemmatizer: their ,    With POS=a: their
Lemmatizer: effect ,    With POS=a: effect
Lemmatizer: . ,    With POS=a: .
Lemmatizer: Lasso ,    With POS=a: Lasso
Lemmatizer: regression ,    With POS=a: regression
Lemmatizer: Lasso ,    With POS=a: Lasso
Lemmatizer: regression ,    With POS=a: regression
Lemmatizer: us ,    With POS=a: uses
Lemmatizer: the ,    With POS=a: the
Lemmatizer: L1 ,    With POS=a: L1
Lemmatizer: penalty ,    With POS=a: penalty
Lemmatizer: term ,    With POS=a: term
Lemmatizer: and ,    With POS=a: and
```

Trigrams

```
('a', 'model', 'but')
('model', 'but', 'does')
('but', 'does', 'not')
('does', 'not', 'reduce')
('not', 'reduce', 'the')
('reduce', 'the', 'number')
('the', 'number', 'of')
('number', 'of', 'variables')
('of', 'variables', ',')
('variables', ',', 'it')
(',', 'it', 'rather')
('it', 'rather', 'just')
('rather', 'just', 'shrinks')
('just', 'shrinks', 'their')
('shrinks', 'their', 'effect')
('their', 'effect', '.')
('effect', '.', 'Lasso')
('.', 'Lasso', 'regression')
('Lasso', 'regression', 'Lasso')
('regression', 'Lasso', 'regression')
('Lasso', 'regression', 'uses')
('regression', 'uses', 'the')
('uses', 'the', 'L1')
('the', 'L1', 'penalty')
('L1', 'penalty', 'term')
('penalty', 'term', 'and')
```

```
Top 10 Repeated Trigrams
 [(('we', 'need', 'to'), 3), (('the', 'coefficients', '.'), 3), (('?',
'?', '='), 3), (('to', 'find', 'the'), 2), (('find', 'the', 'optimal'),
2), (('over', 'a', 'number'), 2), (('a', 'number', 'of'), 2), (('to',
'each', 'other'), 2), (('penalty', 'term', 'to'), 2), (('term', 'to',
'the'), 2)]
```

Sentences pulled using top 10 Repeated Trigrams

```
--> First we need to understand the basics of regression and what
parameters of the equation are changed when using a specific model.
--> But the data we need to define and analyze is not always so easy to
characterize with the base OLS model.
--> = 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.
--> L2 regularization penalty term
The L2 term is equal to the square of the magnitude of the coefficients.
--> is zero then the equation is the basic OLS but if it is greater than
zero then we add a constraint to the coefficients.
--> Here we perform a cross validation and take a peek at the lambda
value corresponding to the lowest prediction error before fitting the
data to the model and viewing the coefficients.
--> 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.
--> The gradient descent algorithm is used to find the optimal cost
function by going over a number of iterations.
--> Visualization of the squared error (from Setosa.io)
```

```
#################### Concatation of Sentences ######################
  First we need to understand the basics of regression and what parameters of the equation are changed when
using a specific model. But the data we need to define and analyze is not always so easy to characterize
with the base OLS model. = 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. L2 regularization penalty term
The L2 term is equal to the square of the magnitude of the coefficients. is zero then the equation is the
basic OLS but if it is greater than zero then we add a constraint to the coefficients. Here we perform a
cross validation and take a peek at the lambda value corresponding to the lowest prediction error before
fitting the data to the model and viewing the coefficients. 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. The gradient descent algorithm is used to find the optimal cost function by
going over a number of iterations. 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. The gradient descent algorithm is used to find the optimal cost function by
going over a number of iterations. The gradient descent algorithm is used to find the optimal cost function
by going over a number of iterations. 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. The gradient descent algorithm is used to
find the optimal cost function by going over a number of iterations. 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. 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. We can see that the R mean-squared values using all three models
```

Task 8:

Editor - C:\Users\geeta\OneDrive\Desktop\python lab\lab1_q8.py

lab1_q2.py | lab1_q3.py* | lab1_q4.py | lab1_q5.py | lab1_q6.py | lab1_q7.py* | lab1_q8.py

```python
# Multiple Linear Regression

# Importing the libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt


# Importing the dataset
dataset = pd.read_csv('50_Startups.csv')

#Checking Nulls
nulls = pd.DataFrame(dataset.isnull().sum().sort_values(ascending=False)[:5])
nulls.columns  = ['Null Count']
nulls.index.name  = 'Feature'
print(nulls)

#Correlation
numeric_features  = dataset.select_dtypes(include=[np.number])
corr = numeric_features.corr()
print("Correlation\n",corr['Profit'].sort_values(ascending=False)[1:4],'\n')

#Splitting Independant and Dependant
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 4].values

#Checking Skew
print("Skew\n",dataset.Profit.skew())
plt.hist(dataset.Profit)
plt.show()

# Encoding categorical data
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
labelencoder = LabelEncoder()
X[:, 3] = labelencoder.fit_transform(X[:, 3])
onehotencoder = OneHotEncoder(categorical_features = [3])
X = onehotencoder.fit_transform(X).toarray()

# Avoiding the Dummy Variable Trap
X = X[:, 1:]

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
```

Editor - C:\Users\geeta\OneDrive\Desktop\python lab\lab1_q8.py

lab1_q2.py | lab1_q3.py* | lab1_q4.py | lab1_q5.py | lab1_q6.py | lab1_q7.py* | lab1_q8.py

```python
nulls = pd.DataFrame(dataset.isnull().sum().sort_values(ascending=False)[:5])
nulls.columns  = ['Null Count']
nulls.index.name  = 'Feature'
print(nulls)

#Correlation
numeric_features  = dataset.select_dtypes(include=[np.number])
corr = numeric_features.corr()
print("Correlation\n",corr['Profit'].sort_values(ascending=False)[1:4],'\n')

#Splitting Independant and Dependant
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 4].values

#Checking Skew
print("Skew\n",dataset.Profit.skew())
plt.hist(dataset.Profit)
plt.show()

# Encoding categorical data
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
labelencoder = LabelEncoder()
X[:, 3] = labelencoder.fit_transform(X[:, 3])
onehotencoder = OneHotEncoder(categorical_features = [3])
X = onehotencoder.fit_transform(X).toarray()

# Avoiding the Dummy Variable Trap
X = X[:, 1:]

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)

# Fitting Multiple Linear Regression to the Training set
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)

# Predicting the Test set results
y_pred = regressor.predict(X_test)

##Evaluate the performance
print ("R^2 is: \n", regressor.score(X_test, y_test))

from sklearn.metrics import mean_squared_error
print ('RMSE is: \n', mean_squared_error(y_test, y_pred))
```

Output:

Dataset to pull 50 startups

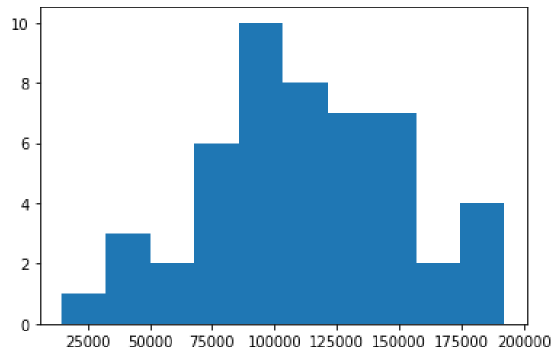
Index	R&D Spend	Administration	Marketing Spend	State	Profit
0	165349	136898	471784	New York	192262
1	162598	151378	443899	California	191792
2	153442	101146	407935	Florida	191050
3	144372	118672	383200	New York	182902
4	142107	91391.8	366168	Florida	166188
5	131877	99814.7	362861	New York	156991
6	134615	147199	127717	California	156123
7	130298	145530	323877	Florida	155753
8	120543	148719	311613	New York	152212
9	123335	108679	304982	California	149760
10	101913	110594	229161	Florida	146122
11	100672	91790.6	249745	California	144259
12	93863.8	127320	249839	Florida	141586
13	91992.4	135495	252665	California	134307
14	119943	156547	256513	Florida	132603
15	114524	122617	261776	New York	129917
16	78013.1	121598	264346	California	126993
17	94657.2	145078	282574	New York	125370
18	91749.2	114176	294920	Florida	124267
19	86419.7	153514	0	New York	122777
20	76253.9	113867	298664	California	118474

Converting state categorical variable into numerical.

	0	1	2	3	4
0	0	1	165349	136898	471784
1	0	0	162598	151378	443899
2	1	0	153442	101146	407935
3	0	1	144372	118672	383200
4	1	0	142107	91391.8	366168
5	0	1	131877	99814.7	362861
6	0	0	134615	147199	127717
7	1	0	130298	145530	323877
8	0	1	120543	148719	311613
9	0	0	123335	108679	304982
10	1	0	101913	110594	229161
11	0	0	100672	91790.6	249745
12	1	0	93863.8	127320	249839
13	0	0	91992.4	135495	252665
14	1	0	119943	156547	256513
15	0	1	114524	122617	261776
16	0	0	78013.1	121598	264346
17	0	1	94657.2	145078	282574
18	1	0	91749.2	114176	294920

Checking the skew.

```
Skew
 0.023291019769116614
```



R^2 and Root Mean Squared Error.

```
R^2 is:
 0.9449726033964256
RMSE is:
 0.7037407490115752
```