Waqas Rasheed Khan
2295250
Logbook
Principles of Data Mining and Machine Learning
School of Computing and Information Sciences

LAB 1

The pandas library, in Python, provides a describe() function that gives an overview of the statistics for a DataFrame. It offers details like count, standard deviation, minimum and maximum values, for data. This functionality allows us to analyze and understand the distribution of data.

In [8]:
```
df.describe()
```

Out[8]:

	Account length	Area code	Number vmail messages	Total day minutes	Total day calls	Total day charge	Total eve minutes	Total eve calls	Total eve charge	Total night minutes	Total night calls	Total night charge	Total intl minutes	Total intl calls	Total intl charge	Customer service calls	(
count	3333.00	3333.00	3333.00	3333.00	3333.00	3333.00	3333.00	3333.00	3333.00	3333.00	3333.00	3333.00	3333.00	3333.00	3333.00	3333.00	33
mean	101.06	437.18	8.10	179.78	100.44	30.56	200.98	100.11	17.08	200.87	100.11	9.04	10.24	4.48	2.76	1.56	
std	39.82	42.37	13.69	54.47	20.07	9.26	50.71	19.92	4.31	50.57	19.57	2.28	2.79	2.46	0.75	1.32	
min	1.00	408.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	23.20	33.00	1.04	0.00	0.00	0.00	0.00	
25%	74.00	408.00	0.00	143.70	87.00	24.43	166.60	87.00	14.16	167.00	87.00	7.52	8.50	3.00	2.30	1.00	
50%	101.00	415.00	0.00	179.40	101.00	30.50	201.40	100.00	17.12	201.20	100.00	9.05	10.30	4.00	2.78	1.00	
75%	127.00	510.00	20.00	216.40	114.00	36.79	235.30	114.00	20.00	235.30	113.00	10.59	12.10	6.00	3.27	2.00	
max	243.00	510.00	51.00	350.80	165.00	59.64	363.70	170.00	30.91	395.00	175.00	17.77	20.00	20.00	5.40	9.00	

Read file

```
df = pd.read_csv("telecom_churn.csv")
df.head()
```

In [2]:
```
df=pd.read_csv("telecom_churn.csv")
df.head()
```

Out[2]:

	State	Account length	Area code	International plan	Voice mail plan	Number vmail messages	Total day minutes	Total day calls	Total day charge	Total eve minutes	Total eve calls	Total eve charge	Total night minutes	Total night calls	Total night charge	Total intl minutes	Total intl calls	Total intl charge	Cust se
0	KS	128	415	No	Yes	25	265.1	110	45.07	197.4	99	16.78	244.7	91	11.01	10.0	3	2.70	
1	OH	107	415	No	Yes	26	161.6	123	27.47	195.5	103	16.62	254.4	103	11.45	13.7	3	3.70	
2	NJ	137	415	No	No	0	243.4	114	41.38	121.2	110	10.30	162.6	104	7.32	12.2	5	3.29	
3	OH	84	408	Yes	No	0	299.4	71	50.90	61.9	88	5.26	196.9	89	8.86	6.6	7	1.78	
4	OK	75	415	Yes	No	0	166.7	113	28.34	148.3	122	12.61	186.9	121	8.41	10.1	3	2.73	

In [8]:
```python
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3333 entries, 0 to 3332
Data columns (total 20 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   State                   3333 non-null   object 
 1   Account length          3333 non-null   int64  
 2   Area code               3333 non-null   int64  
 3   International plan      3333 non-null   object 
 4   Voice mail plan         3333 non-null   object 
 5   Number vmail messages   3333 non-null   int64  
 6   Total day minutes       3333 non-null   float64
 7   Total day calls         3333 non-null   int64  
 8   Total day charge        3333 non-null   float64
 9   Total eve minutes       3333 non-null   float64
 10  Total eve calls         3333 non-null   int64  
 11  Total eve charge        3333 non-null   float64
 12  Total night minutes     3333 non-null   float64
 13  Total night calls       3333 non-null   int64  
 14  Total night charge      3333 non-null   float64
 15  Total intl minutes      3333 non-null   float64
 16  Total intl calls        3333 non-null   int64  
 17  Total intl charge       3333 non-null   float64
 18  Customer service calls  3333 non-null   int64  
 19  Churn                   3333 non-null   bool   
dtypes: bool(1), float64(8), int64(8), object(3)
memory usage: 498.1+ KB
None
```

You have the ability to organize a dataset according to the value of one of its variables, such, as columns. For example you can sort it by day charge by utilizing ascending=False to arrange it in descending order.

In [17]:
```python
df.sort_values(by="Total day charge", ascending = False).head()
```

Out[17]:

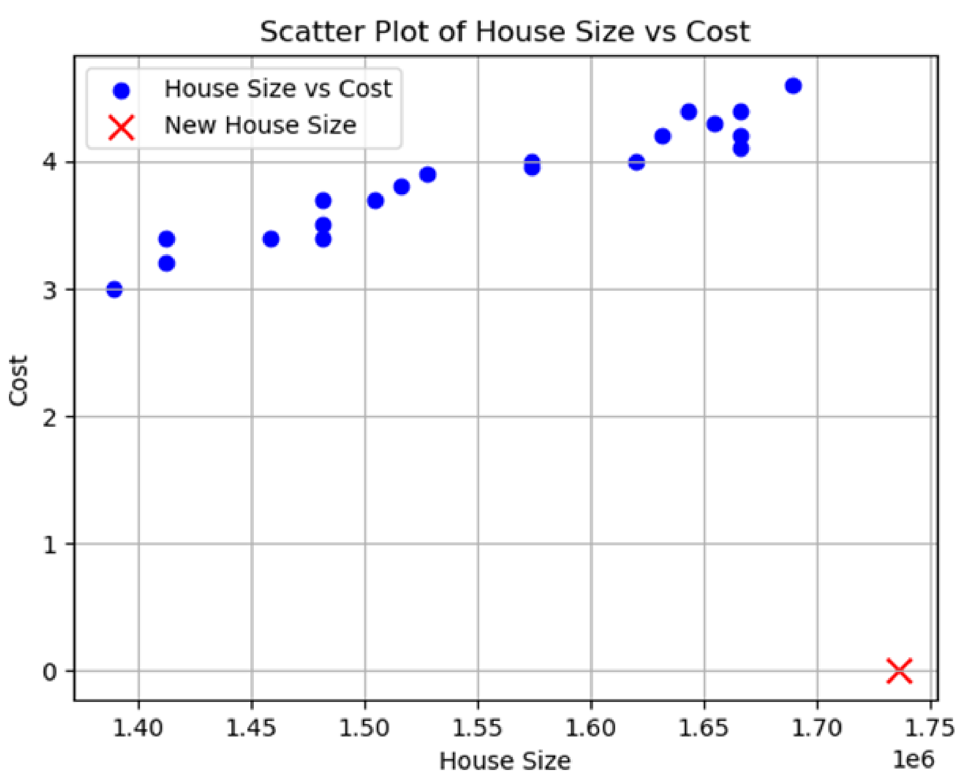
	State	Account length	Area code	International plan	Voice mail plan	Number vmail messages	Total day minutes	Total day calls	Total day charge	Total eve minutes	Total eve calls	Total eve charge	Total night minutes	Total night calls	Total night charge	Total intl minutes	Total intl calls	Total intl charge	C
365	CO	154	415	No	No	0	350.8	75	59.64	216.5	94	18.40	253.9	100	11.43	10.1	9	2.73	
985	NY	64	415	Yes	No	0	346.8	55	58.96	249.5	79	21.21	275.4	102	12.39	13.3	9	3.59	
2594	OH	115	510	Yes	No	0	345.3	81	58.70	203.4	106	17.29	217.5	107	9.79	11.8	8	3.19	
156	OH	83	415	No	No	0	337.4	120	57.36	227.4	116	19.33	153.9	114	6.93	15.8	7	4.27	
605	MO	112	415	No	No	0	335.5	77	57.04	212.5	109	18.06	265.0	132	11.93	12.7	8	3.43	

In Python the pandas library offers a function called value_counts(). This function allows you to determine how times each distinct value occurs in a Series.

```
In [14]: df["International plan"].value_counts()

Out[14]: No     3010
         Yes     323
         Name: International plan, dtype: int64
```

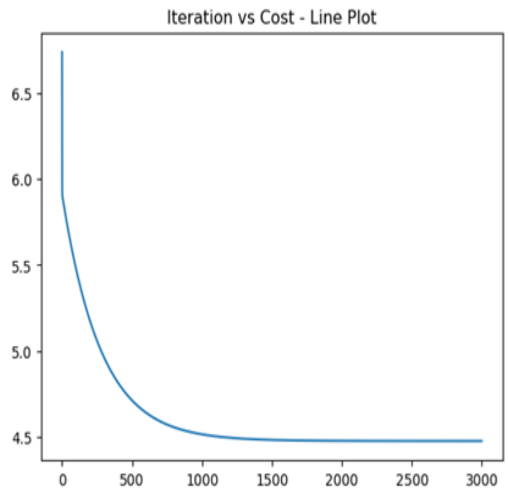
Lab 2

A Scatter plot is commonly used to show the correlation between the size of a house and its cost. The Y Axis represents the cost while the X Axis indicates the size of the house. It can be observed that as the size of a house increases its cost also tends to increase.



Lab 3

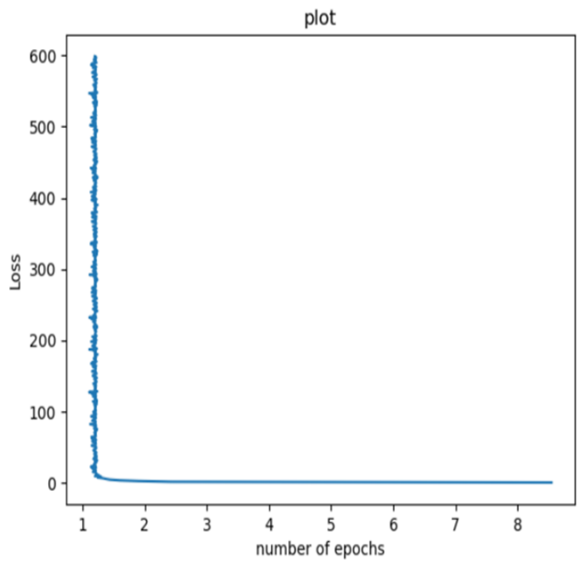
```
In [22]: # Your code to plot all costs
         plt.plot(J_history)
         plt.title('Iteration vs Cost - Line Plot')

Out[22]: Text(0.5, 1.0, 'Iteration vs Cost - Line Plot')
```




```
In [37]: SID = 2295250
         First_City = SID/10 # Put the population of first city as 10 times less than your SID
         Second_City = SID/30 # Put the population of second city as 30 times less than your SID

         predict1 = (prediction(([1, First_City/10000]),(new_theta)))
         predict2 = (prediction(([1, Second_City/10000]),(new_theta)))

         print(f'For a population of {First_City} people, profit will be {predict1[0]} ')
         print(f'For a population of {Second_City} people, profit will be {predict2[0]}')

         For a population of 229525.0 people, profit will be 234641.7190432843
         For a population of 76508.33333333333 people, profit will be 52360.23184742697
```

Lab 4

In [88]:
```
# Your code to plot epochs vs loss. Call the method.
plotLoss(num_epochs,train_loss)
```



In [89]:
```
#Code to predict the profit i.e. y values
predict(theta_updated,y_train)
```

Out[89]: -4.797723428027043

Lab 5

Calculating accuracy - number of correct classification/total number of classification.

In [42]:
```
np.mean(predict(res.x, X) == y)
```

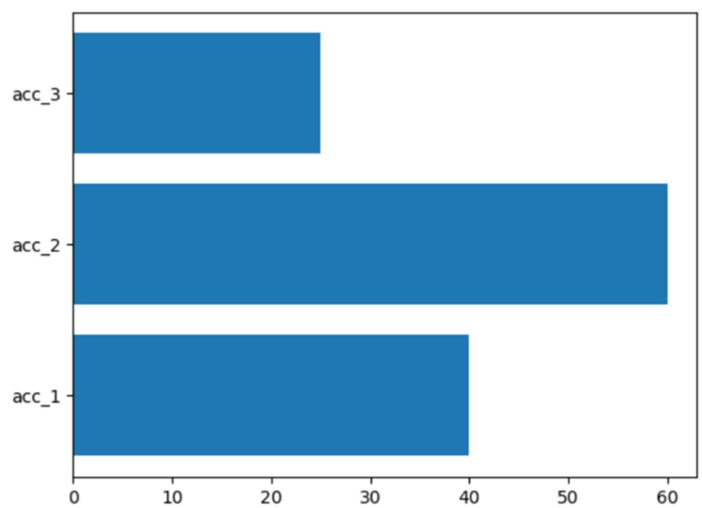
Out[42]: 0.89

```
In [22]: resultant_accuracy = 0.89
         SID = 2295250
         encrypted_value = resultant_accuracy*SID
         print(encrypted_value)

         2042772.5
```

Lab 6

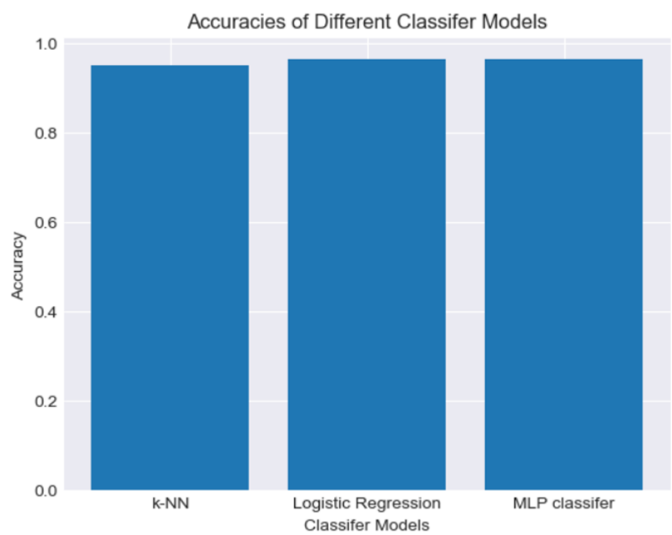
In my analysis I have evaluated the precision of MLP models using the class. I considered three accuracy levels, 40%, 60% and 25%.

```
In [17]: #Assuming accuracy - 40, 60, 25
         plt.barh(['acc_1', 'acc_2', 'acc_3'], [40, 60, 25])

Out[17]: <BarContainer object of 3 artists>
```

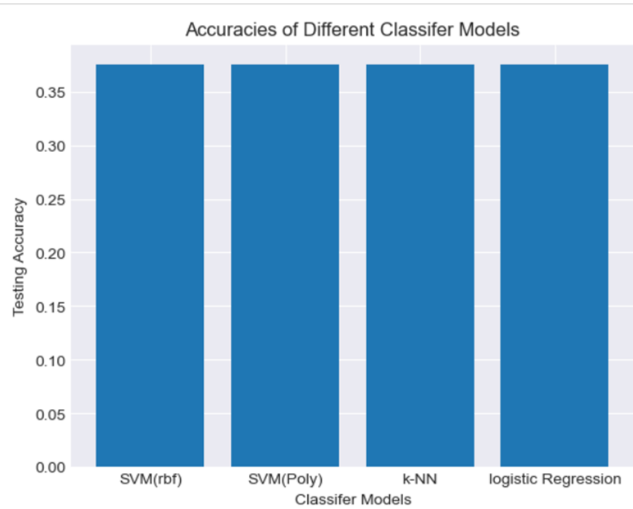


Lab 8

In this case a bar chart is employed to showcase the variation, in accuracies across classifier models such, as KNN, Logistic Regression and MLP Classifier. It appears that there is no disparity observed among these classifier models.



Lab 9



According to the confusion matrix logistic regression demonstrates the performance while KNN performs the least effectively.

Lab 10

Q1:

At each node of the decision tree the criteria used to make decisions are information gain and Gini impurity. These criteria help determine the quality of test decisions and how they classify samples into classes.

Q2:

In a decision tree the measure of disorder such as entropy, Gini index or loss decreases at each node when the split leads to subsets that are more similar. The objective is to reduce disorder (entropy or Gini index) or loss in order to improve the tree's ability to differentiate or make predictions about the target variable.

Q3:

As we descend further into the tree it is common for the values of Entropy/Gini/Loss Change to decrease.

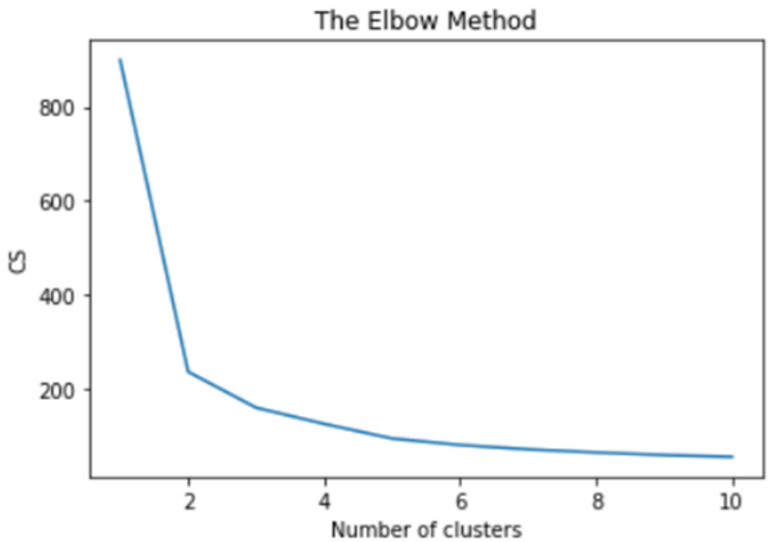
Q4:

Typically the quantity of data samples at each node in a tree building process is influenced by the data itself and the splits that are made. A general trend is that nodes representing categories or conditions which are closer to the root tend to have a higher number of samples.

Q5:

In a decision tree when we reach a leaf node it usually contains information about the predicted output or class for the subset of data that led to that leaf. In classification tasks the leaf node holds the class label assigned to the majority of samples, within that subset.

Lab 11




```
In [38]: kmeans = KMeans(n_clusters=3, random_state=0)

kmeans.fit(X)

# check how many of the samples were correctly labeled
labels = kmeans.labels_

correct_labels = sum(y == labels)
print("Result: %d out of %d samples were correctly labeled." % (correct_labels, y.size))
print('Accuracy score: {0:0.2f}'. format(correct_labels/float(y.size)))
```

```
C:\Users\asimq\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will c
hange from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```

```
Result: 4165 out of 7050 samples were correctly labeled.
Accuracy score: 0.59
```

```
In [ ]: kmeans = KMeans(n_clusters=4, random_state=0)

kmeans.fit(X)

# check how many of the samples were correctly labeled
labels = kmeans.labels_

correct_labels = sum(y == labels)
print("Result: %d out of %d samples were correctly labeled." % (correct_labels, y.size))
print('Accuracy score: {0:0.2f}'. format(correct_labels/float(y.size)))
```

```
Result: 4340 out of 7050 samples were correctly labeled.
Accuracy score: 0.62
```