DATA ANALYTICS

THEORY ASSIGNMENT 1

NAME-ADITYA SHARMA.A

ROLL NUMBER-22MCB0014

DATASET- SHUTTLE-LANDING-CONTROL.csv

About the Dataset

1. Title: Space Shuttle Autolanding Domain

2. Sources:
   (a) Original source: unknown
      -- NASA: Mr. Roger Burke's autolander design team
   (b) Donor: Bojan Cestnik
         Jozef Stefan Institute
         Jamova 39
         61000 Ljubljana
         Yugoslavia (tel.: (38)(+61) 214-399 ext.287)
   (c) Date: November 1988

3. Past Usage: (several, it appears)
   Example: Michie,D. (1988). The Fifth Generation's Unbridged Gap.
         In Rolf Herken (Ed.) The Universal Turing Machine: A
         Half-Century Survey, 466-489, Oxford University Press.

4. Relevant Information:
   This is a tiny database. Michie reports that Burke's group used
   RULEMASTER to generate comprehendable rules for determining
   the conditions under which an autolanding would be preferable to
   manual control of the spacecraft.

5. Number of Instances: 15

6. Number of Attributes: 7 (including the class attribute)

7. Attribute Information:

1. Class: noauto, auto
   -- that is, advise using manual/automatic control
2. STABILITY: stab, xstab
3. ERROR: XL, LX, MM, SS
4. SIGN: pp, nn
5. WIND: head, tail
6. MAGNITUDE: Low, Medium, Strong, OutOfRange
7. VISIBILITY: yes, no

8. Missing Attribute Values:
   -- none
   -- but several "don't care" values: (denoted by "*")

Attribute Number:	Number of Don't Care Values:
2:	2
3:	3
4:	8
5:	8
6:	5
7:	0

9. Class Distribution:
   1. Use noauto control: 6
   2. Use automatic control:

Implementation-

1. Importing the libraries

```
import numpy as np
import pandas as pd
```

2.Importing the dataset

```
[19] df=pd.read_csv('/content/csv_result-dataset_114_shuttle-landing-control.csv')
```

3.Basic Data Exploration

```
#Basic Data Exploration
print("Number of rows:", len(df))
print("Number of columns:", len(df.columns))
print("Data types of each column:")
print(df.dtypes)
print("Missing values:")
print(df.isnull().sum())
```

```
Number of rows: 15
Number of columns: 8
Data types of each column:
id              int64
Class           int64
STABILITY      object
ERROR          object
SIGN           object
WIND           object
MAGNITUDE      object
VISIBILITY      int64
dtype: object
Missing values:
id             0
Class          0
STABILITY      0
ERROR          0
SIGN           0
WIND           0
MAGNITUDE      0
VISIBILITY     0
dtype: int64
```

4.Handling of missing values

Here,we fill up the missing values by using the simple imputer .The numerical meathod we use is the mode or most frequent occurance is filled in the missing slots

```
[48] #Handling the missing values
     from sklearn.impute import SimpleImputer
     imputer = SimpleImputer(strategy='most_frequent')
     df_filled = pd.DataFrame(imputer.fit_transform(df), columns=df.columns)
     print(df_filled)
```

	id	Class	STABILITY	ERROR	SIGN	WIND	MAGNITUDE	VISIBILITY
0	3	1	1	2	?	?	?	1
1	4	1	1	1	?	?	?	1
2	5	1	1	3	2	2	?	1
3	7	2	1	4	?	?	1	1
4	8	2	1	4	?	?	2	1
5	9	2	1	4	?	?	3	1
6	10	2	1	3	1	1	1	1
7	11	2	1	3	1	1	2	1
8	12	2	1	3	1	2	1	1
9	13	2	1	3	1	2	2	1
10	14	1	1	3	1	1	3	1
11	15	2	1	3	1	2	3	1

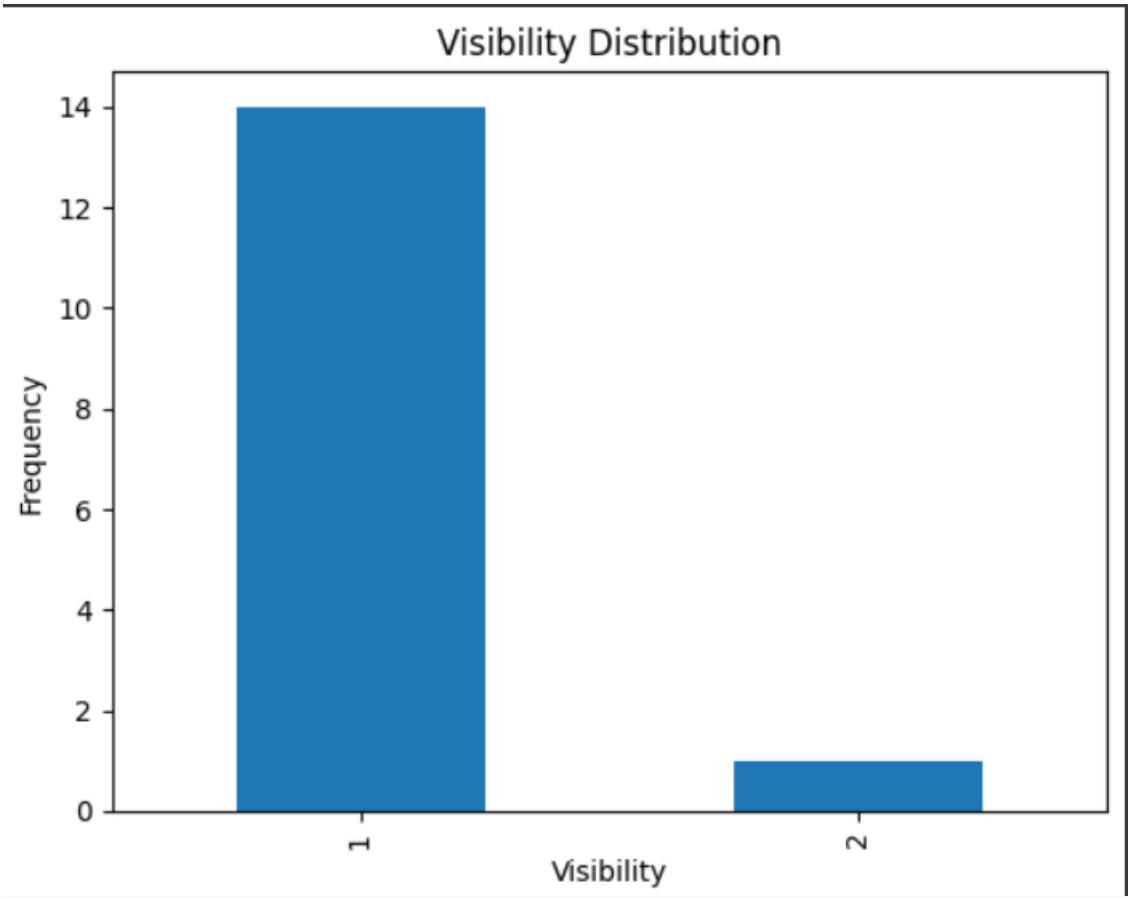
5. Descriptive analysis

Here, we use the df.describe() to show the descriptive analysis

```
#Descriptive analysis
print("Summary statistics:")
print(df.describe())
```

```
Summary statistics:
              id      Class  VISIBILITY
count  15.000000  15.000000   15.000000
mean    8.000000   1.600000    1.066667
std     4.472136   0.507093    0.258199
min     1.000000   1.000000    1.000000
25%     4.500000   1.000000    1.000000
50%     8.000000   2.000000    1.000000
75%    11.500000   2.000000    1.000000
max    15.000000   2.000000    2.000000
```

Next, we plot a frequency graph for the visibility



6. Printing the correlation matrix

```
#PRINTING THE CORRELATION MATRIX
correlation_matrix = df.corr()
print("Correlation matrix:")
print(correlation_matrix)
```

```
Correlation matrix:
                   id     Class  VISIBILITY
id          1.000000  0.440959   -0.433013
Class       0.440959  1.000000    0.218218
VISIBILITY -0.433013  0.218218    1.000000
<ipython-input-25-7a7341cd5616>:1: FutureWarning: The default value of numeric_or
  correlation_matrix = df.corr()
```

7. Performing predictive analysis using logistic regression and seeing the accuracy score

```
#Performing Predictive analysis on ERROR Using Logistic Regression
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

target_variable = 'ERROR'  # Replace 'ERROR' with the desired column name as the target variable

# Remove rows with '?' in the target variable column
df = df[df[target_variable] != '?']

# Convert target variable to numeric
df[target_variable] = pd.to_numeric(df[target_variable])

# Separate features and target variable
X = df.drop([target_variable], axis=1)  # Features
y = df[target_variable]  # Target variable

# Perform one-hot encoding on categorical variables
X = pd.get_dummies(X)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = LogisticRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.6666666666666666
```

8.Drawing a KMeans Clustering Graph using K=3

```
#Drawing a Kmeans Clustering Graph with K=3
from sklearn.cluster import KMeans



target_variable = 'ERROR'  # Replace 'ERROR' with the desired column name as the target variable

# Remove rows with '?' in the target variable column
df_filled = df_filled[df_filled[target_variable] != '?']

# Convert target variable to numeric
df_filled[target_variable] = pd.to_numeric(df_filled[target_variable])

target_variable = 'VISIBILITY'  # Replace 'ERROR' with the desired column name as the target variable

# Remove rows with '?' in the target variable column
df_filled = df_filled[df_filled[target_variable] != '?']

# Convert target variable to numeric
df_filled[target_variable] = pd.to_numeric(df_filled[target_variable])


features = df_filled[['VISIBILITY', 'ERROR']]  # Features for clustering

# Define the number of clusters
num_clusters = 3
```

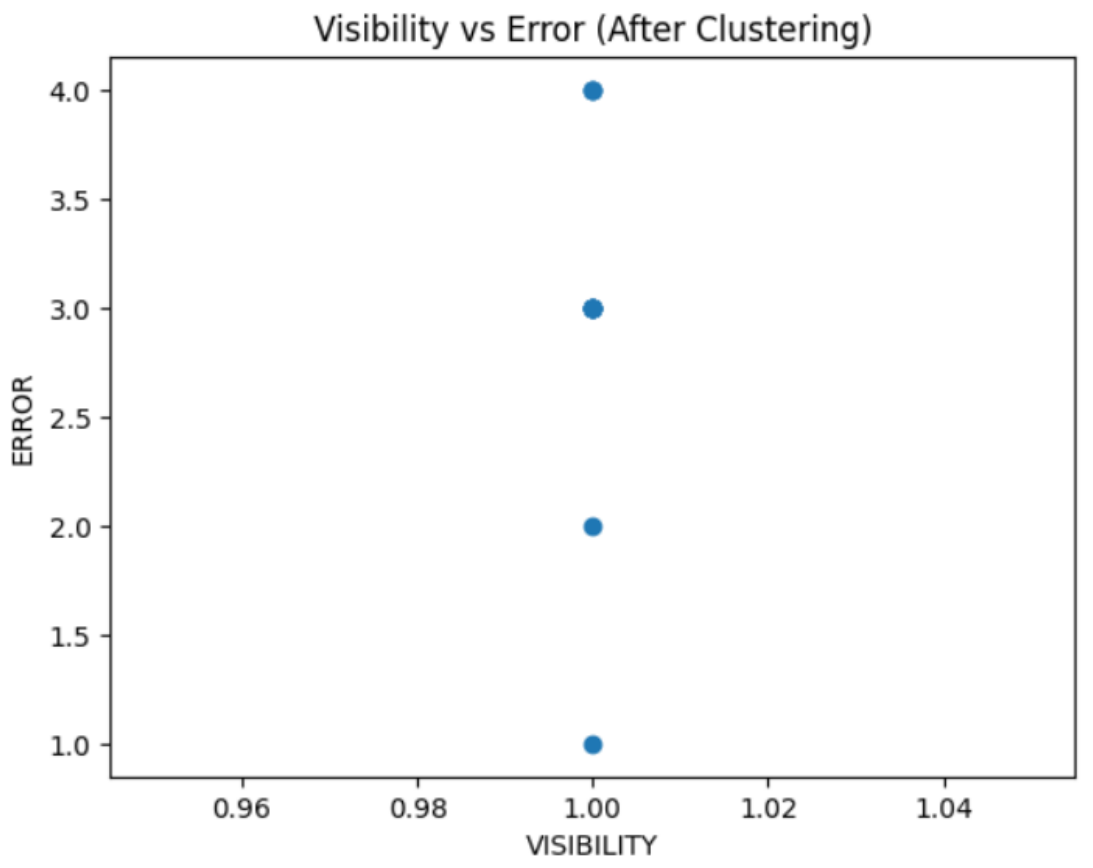
```python
features = df_filled[['VISIBILITY', 'ERROR']]  # Features for clustering

# Define the number of clusters
num_clusters = 3

kmeans = KMeans(n_clusters=num_clusters)
kmeans.fit(features)
cluster_labels = kmeans.labels_

# Add the cluster labels to the dataset
df['Cluster'] = cluster_labels

# Step 7: Visualization of 'VISIBILITY' vs 'ERROR' after clustering
plt.scatter(df_filled['VISIBILITY'], df_filled['ERROR'])
plt.xlabel('VISIBILITY')
plt.ylabel('ERROR')
plt.title('Visibility vs Error (After Clustering)')
plt.show()
```



Inference- By analyzing the given dataset we can conclude that the error in a shuttle landing is not dependent on the visibility on that day.This is because modern shuttles are equipped with radar,GPS and other such high tech equipments.