BITS F464 - Semester 1 - MACHINE LEARNING

ASSIGNMENT 2 – DECISION TREES AND SUPPORT VECTOR MACHINES

-Please rename the file as "TeamXX_Assignment2.ipynb"

Team number: 31

(*In Title case, separated with commas*) ***Full names of all students in the team:***

Kushal Chakraborty,Ajinkya Medhekar, Ashutosh Wagh, S Shashank, Srinidhi P Katte

(*Separated by commas*) ***Id number of all students in the team:*** 2022H1030089H, 2022H1030099H, 2022H1030052H, 2022H1030067H, 2022H1030075H

This assignment aims to identify the differences between three Machine Learning models.

1. Preprocess and perform exploratory data analysis of the dataset obtained

In [1]:
```
import numpy as np
from pandas import read_csv
import matplotlib.pyplot as plt
from pandas import DataFrame
from sklearn.impute import SimpleImputer
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report,confusion_matrix
from sklearn.metrics import roc_auc_score
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
import pandas as pd
import time
import numpy
import pandas
import random
import plotly.graph_objects as go
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
```

```
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
features = read_csv('attributes.csv', delim_whitespace = True)
dataset_dataframe = read_csv('communities.data', names = features['attributes'])
```

In [3]:
```
dataset_dataframe = dataset_dataframe.drop(columns=['state','county','community','c
dataset_dataframe.head()
```

Out[3]:

	population	householdsize	racepctblack	racePctWhite	racePctAsian	racePctHisp	ageP
0	0.19	0.33	0.02	0.90	0.12	0.17	
1	0.00	0.16	0.12	0.74	0.45	0.07	
2	0.00	0.42	0.49	0.56	0.17	0.04	
3	0.04	0.77	1.00	0.08	0.12	0.10	
4	0.01	0.55	0.02	0.95	0.09	0.05	

5 rows × 123 columns

Observation

We have deleted some features which are not very useful for models in order to predict the labels

In [4]:
```
dataset_dataframe = dataset_dataframe.replace('?', np.nan)
features_missing_values = dataset_dataframe.columns[dataset_dataframe.isnull().any(
```

In [5]:
```
print(features_missing_values)
features_missing_values.shape
```

```
Index(['OtherPerCap', 'LemasSwornFT', 'LemasSwFTPerPop', 'LemasSwFTFieldOps',
       'LemasSwFTFieldPerPop', 'LemasTotalReq', 'LemasTotReqPerPop',
       'PolicReqPerOffic', 'PolicPerPop', 'RacialMatchCommPol',
       'PctPolicWhite', 'PctPolicBlack', 'PctPolicHisp', 'PctPolicAsian',
       'PctPolicMinor', 'OfficAssgnDrugUnits', 'NumKindsDrugsSeiz',
       'PolicAveOTWorked', 'PolicCars', 'PolicOperBudg', 'LemasPctPolicOnPatr',
       'LemasGangUnitDeploy', 'PolicBudgPerPop'],
      dtype='object')
```

Out[5]: (23,)

Observation

There are 23 features which have missing values.

In [6]:
```
replace_missingval_with_mean = SimpleImputer(missing_values=np.nan, strategy='mean'
replace_missingval_with_mean = replace_missingval_with_mean.fit(dataset_dataframe[[
dataset_dataframe[['OtherPerCap']] = replace_missingval_with_mean.transform(dataset
dataset_dataframe = dataset_dataframe.dropna(axis=1)
```

```
print(dataset_dataframe.shape)
dataset_dataframe.head()
```

```
(1994, 101)
```

Out[6]:

	population	householdsize	racepctblack	racePctWhite	racePctAsian	racePctHisp	ageP
0	0.19	0.33	0.02	0.90	0.12	0.17	
1	0.00	0.16	0.12	0.74	0.45	0.07	
2	0.00	0.42	0.49	0.56	0.17	0.04	
3	0.04	0.77	1.00	0.08	0.12	0.10	
4	0.01	0.55	0.02	0.95	0.09	0.05	

5 rows × 101 columns

Observation

We have replaced the missing values in each feature with the mean of all the values contained within them.

In [7]:
```
X = dataset_dataframe.iloc[:, 0:100].values
y = dataset_dataframe.iloc[:, 100].values
```

In [8]:
```
frequency_dict = {}

# Iterate through the specified ranges
for i in range(0, 1000):
    lower_bound = i / 10
    upper_bound = (i + 1) / 10
    count = sum(1 for num in y if lower_bound <= num < upper_bound)
    frequency_dict[f'{lower_bound}-{upper_bound}'] = count

# Print the frequencies
for key, value in frequency_dict.items():
    print(f'Frequency of numbers in range {key}: {value}')
```

```
Frequency of numbers in range 0.0-0.1: 679
Frequency of numbers in range 0.1-0.2: 470
Frequency of numbers in range 0.2-0.3: 285
Frequency of numbers in range 0.3-0.4: 174
Frequency of numbers in range 0.4-0.5: 97
Frequency of numbers in range 0.5-0.6: 98
Frequency of numbers in range 0.6-0.7: 69
Frequency of numbers in range 0.7-0.8: 32
Frequency of numbers in range 0.8-0.9: 33
Frequency of numbers in range 0.9-1.0: 13
Frequency of numbers in range 1.0-1.1: 44
Frequency of numbers in range 1.1-1.2: 0
Frequency of numbers in range 1.2-1.3: 0
Frequency of numbers in range 1.3-1.4: 0
Frequency of numbers in range 1.4-1.5: 0
Frequency of numbers in range 1.5-1.6: 0
```

```
Frequency of numbers in range 1.6-1.7: 0
Frequency of numbers in range 1.7-1.8: 0
Frequency of numbers in range 1.8-1.9: 0
Frequency of numbers in range 1.9-2.0: 0
Frequency of numbers in range 2.0-2.1: 0
Frequency of numbers in range 2.1-2.2: 0
Frequency of numbers in range 2.2-2.3: 0
Frequency of numbers in range 2.3-2.4: 0
Frequency of numbers in range 2.4-2.5: 0
Frequency of numbers in range 2.5-2.6: 0
Frequency of numbers in range 2.6-2.7: 0
Frequency of numbers in range 2.7-2.8: 0
Frequency of numbers in range 2.8-2.9: 0
Frequency of numbers in range 2.9-3.0: 0
Frequency of numbers in range 3.0-3.1: 0
Frequency of numbers in range 3.1-3.2: 0
Frequency of numbers in range 3.2-3.3: 0
Frequency of numbers in range 3.3-3.4: 0
Frequency of numbers in range 3.4-3.5: 0
Frequency of numbers in range 3.5-3.6: 0
Frequency of numbers in range 3.6-3.7: 0
Frequency of numbers in range 3.7-3.8: 0
Frequency of numbers in range 3.8-3.9: 0
Frequency of numbers in range 3.9-4.0: 0
Frequency of numbers in range 4.0-4.1: 0
Frequency of numbers in range 4.1-4.2: 0
Frequency of numbers in range 4.2-4.3: 0
Frequency of numbers in range 4.3-4.4: 0
Frequency of numbers in range 4.4-4.5: 0
Frequency of numbers in range 4.5-4.6: 0
Frequency of numbers in range 4.6-4.7: 0
Frequency of numbers in range 4.7-4.8: 0
Frequency of numbers in range 4.8-4.9: 0
Frequency of numbers in range 4.9-5.0: 0
Frequency of numbers in range 5.0-5.1: 0
Frequency of numbers in range 5.1-5.2: 0
Frequency of numbers in range 5.2-5.3: 0
Frequency of numbers in range 5.3-5.4: 0
Frequency of numbers in range 5.4-5.5: 0
Frequency of numbers in range 5.5-5.6: 0
Frequency of numbers in range 5.6-5.7: 0
Frequency of numbers in range 5.7-5.8: 0
Frequency of numbers in range 5.8-5.9: 0
Frequency of numbers in range 5.9-6.0: 0
Frequency of numbers in range 6.0-6.1: 0
Frequency of numbers in range 6.1-6.2: 0
Frequency of numbers in range 6.2-6.3: 0
Frequency of numbers in range 6.3-6.4: 0
Frequency of numbers in range 6.4-6.5: 0
Frequency of numbers in range 6.5-6.6: 0
Frequency of numbers in range 6.6-6.7: 0
Frequency of numbers in range 6.7-6.8: 0
Frequency of numbers in range 6.8-6.9: 0
Frequency of numbers in range 6.9-7.0: 0
Frequency of numbers in range 7.0-7.1: 0
Frequency of numbers in range 7.1-7.2: 0
```

```
Frequency of numbers in range 7.2-7.3: 0
Frequency of numbers in range 7.3-7.4: 0
Frequency of numbers in range 7.4-7.5: 0
Frequency of numbers in range 7.5-7.6: 0
Frequency of numbers in range 7.6-7.7: 0
Frequency of numbers in range 7.7-7.8: 0
Frequency of numbers in range 7.8-7.9: 0
Frequency of numbers in range 7.9-8.0: 0
Frequency of numbers in range 8.0-8.1: 0
Frequency of numbers in range 8.1-8.2: 0
Frequency of numbers in range 8.2-8.3: 0
Frequency of numbers in range 8.3-8.4: 0
Frequency of numbers in range 8.4-8.5: 0
Frequency of numbers in range 8.5-8.6: 0
Frequency of numbers in range 8.6-8.7: 0
Frequency of numbers in range 8.7-8.8: 0
Frequency of numbers in range 8.8-8.9: 0
Frequency of numbers in range 8.9-9.0: 0
Frequency of numbers in range 9.0-9.1: 0
Frequency of numbers in range 9.1-9.2: 0
Frequency of numbers in range 9.2-9.3: 0
Frequency of numbers in range 9.3-9.4: 0
Frequency of numbers in range 9.4-9.5: 0
Frequency of numbers in range 9.5-9.6: 0
Frequency of numbers in range 9.6-9.7: 0
Frequency of numbers in range 9.7-9.8: 0
Frequency of numbers in range 9.8-9.9: 0
Frequency of numbers in range 9.9-10.0: 0
Frequency of numbers in range 10.0-10.1: 0
Frequency of numbers in range 10.1-10.2: 0
Frequency of numbers in range 10.2-10.3: 0
Frequency of numbers in range 10.3-10.4: 0
Frequency of numbers in range 10.4-10.5: 0
Frequency of numbers in range 10.5-10.6: 0
Frequency of numbers in range 10.6-10.7: 0
Frequency of numbers in range 10.7-10.8: 0
Frequency of numbers in range 10.8-10.9: 0
Frequency of numbers in range 10.9-11.0: 0
Frequency of numbers in range 11.0-11.1: 0
Frequency of numbers in range 11.1-11.2: 0
Frequency of numbers in range 11.2-11.3: 0
Frequency of numbers in range 11.3-11.4: 0
Frequency of numbers in range 11.4-11.5: 0
Frequency of numbers in range 11.5-11.6: 0
Frequency of numbers in range 11.6-11.7: 0
Frequency of numbers in range 11.7-11.8: 0
Frequency of numbers in range 11.8-11.9: 0
Frequency of numbers in range 11.9-12.0: 0
Frequency of numbers in range 12.0-12.1: 0
Frequency of numbers in range 12.1-12.2: 0
Frequency of numbers in range 12.2-12.3: 0
Frequency of numbers in range 12.3-12.4: 0
Frequency of numbers in range 12.4-12.5: 0
Frequency of numbers in range 12.5-12.6: 0
Frequency of numbers in range 12.6-12.7: 0
Frequency of numbers in range 12.7-12.8: 0
```

```
Frequency of numbers in range 12.8-12.9: 0
Frequency of numbers in range 12.9-13.0: 0
Frequency of numbers in range 13.0-13.1: 0
Frequency of numbers in range 13.1-13.2: 0
Frequency of numbers in range 13.2-13.3: 0
Frequency of numbers in range 13.3-13.4: 0
Frequency of numbers in range 13.4-13.5: 0
Frequency of numbers in range 13.5-13.6: 0
Frequency of numbers in range 13.6-13.7: 0
Frequency of numbers in range 13.7-13.8: 0
Frequency of numbers in range 13.8-13.9: 0
Frequency of numbers in range 13.9-14.0: 0
Frequency of numbers in range 14.0-14.1: 0
Frequency of numbers in range 14.1-14.2: 0
Frequency of numbers in range 14.2-14.3: 0
Frequency of numbers in range 14.3-14.4: 0
Frequency of numbers in range 14.4-14.5: 0
Frequency of numbers in range 14.5-14.6: 0
Frequency of numbers in range 14.6-14.7: 0
Frequency of numbers in range 14.7-14.8: 0
Frequency of numbers in range 14.8-14.9: 0
Frequency of numbers in range 14.9-15.0: 0
Frequency of numbers in range 15.0-15.1: 0
Frequency of numbers in range 15.1-15.2: 0
Frequency of numbers in range 15.2-15.3: 0
Frequency of numbers in range 15.3-15.4: 0
Frequency of numbers in range 15.4-15.5: 0
Frequency of numbers in range 15.5-15.6: 0
Frequency of numbers in range 15.6-15.7: 0
Frequency of numbers in range 15.7-15.8: 0
Frequency of numbers in range 15.8-15.9: 0
Frequency of numbers in range 15.9-16.0: 0
Frequency of numbers in range 16.0-16.1: 0
Frequency of numbers in range 16.1-16.2: 0
Frequency of numbers in range 16.2-16.3: 0
Frequency of numbers in range 16.3-16.4: 0
Frequency of numbers in range 16.4-16.5: 0
Frequency of numbers in range 16.5-16.6: 0
Frequency of numbers in range 16.6-16.7: 0
Frequency of numbers in range 16.7-16.8: 0
Frequency of numbers in range 16.8-16.9: 0
Frequency of numbers in range 16.9-17.0: 0
Frequency of numbers in range 17.0-17.1: 0
Frequency of numbers in range 17.1-17.2: 0
Frequency of numbers in range 17.2-17.3: 0
Frequency of numbers in range 17.3-17.4: 0
Frequency of numbers in range 17.4-17.5: 0
Frequency of numbers in range 17.5-17.6: 0
Frequency of numbers in range 17.6-17.7: 0
Frequency of numbers in range 17.7-17.8: 0
Frequency of numbers in range 17.8-17.9: 0
Frequency of numbers in range 17.9-18.0: 0
Frequency of numbers in range 18.0-18.1: 0
Frequency of numbers in range 18.1-18.2: 0
Frequency of numbers in range 18.2-18.3: 0
Frequency of numbers in range 18.3-18.4: 0
```

```
Frequency of numbers in range 18.4-18.5: 0
Frequency of numbers in range 18.5-18.6: 0
Frequency of numbers in range 18.6-18.7: 0
Frequency of numbers in range 18.7-18.8: 0
Frequency of numbers in range 18.8-18.9: 0
Frequency of numbers in range 18.9-19.0: 0
Frequency of numbers in range 19.0-19.1: 0
Frequency of numbers in range 19.1-19.2: 0
Frequency of numbers in range 19.2-19.3: 0
Frequency of numbers in range 19.3-19.4: 0
Frequency of numbers in range 19.4-19.5: 0
Frequency of numbers in range 19.5-19.6: 0
Frequency of numbers in range 19.6-19.7: 0
Frequency of numbers in range 19.7-19.8: 0
Frequency of numbers in range 19.8-19.9: 0
Frequency of numbers in range 19.9-20.0: 0
Frequency of numbers in range 20.0-20.1: 0
Frequency of numbers in range 20.1-20.2: 0
Frequency of numbers in range 20.2-20.3: 0
Frequency of numbers in range 20.3-20.4: 0
Frequency of numbers in range 20.4-20.5: 0
Frequency of numbers in range 20.5-20.6: 0
Frequency of numbers in range 20.6-20.7: 0
Frequency of numbers in range 20.7-20.8: 0
Frequency of numbers in range 20.8-20.9: 0
Frequency of numbers in range 20.9-21.0: 0
Frequency of numbers in range 21.0-21.1: 0
Frequency of numbers in range 21.1-21.2: 0
Frequency of numbers in range 21.2-21.3: 0
Frequency of numbers in range 21.3-21.4: 0
Frequency of numbers in range 21.4-21.5: 0
Frequency of numbers in range 21.5-21.6: 0
Frequency of numbers in range 21.6-21.7: 0
Frequency of numbers in range 21.7-21.8: 0
Frequency of numbers in range 21.8-21.9: 0
Frequency of numbers in range 21.9-22.0: 0
Frequency of numbers in range 22.0-22.1: 0
Frequency of numbers in range 22.1-22.2: 0
Frequency of numbers in range 22.2-22.3: 0
Frequency of numbers in range 22.3-22.4: 0
Frequency of numbers in range 22.4-22.5: 0
Frequency of numbers in range 22.5-22.6: 0
Frequency of numbers in range 22.6-22.7: 0
Frequency of numbers in range 22.7-22.8: 0
Frequency of numbers in range 22.8-22.9: 0
Frequency of numbers in range 22.9-23.0: 0
Frequency of numbers in range 23.0-23.1: 0
Frequency of numbers in range 23.1-23.2: 0
Frequency of numbers in range 23.2-23.3: 0
Frequency of numbers in range 23.3-23.4: 0
Frequency of numbers in range 23.4-23.5: 0
Frequency of numbers in range 23.5-23.6: 0
Frequency of numbers in range 23.6-23.7: 0
Frequency of numbers in range 23.7-23.8: 0
Frequency of numbers in range 23.8-23.9: 0
Frequency of numbers in range 23.9-24.0: 0
```

Frequency of numbers in range 24.0-24.1: 0
Frequency of numbers in range 24.1-24.2: 0
Frequency of numbers in range 24.2-24.3: 0
Frequency of numbers in range 24.3-24.4: 0
Frequency of numbers in range 24.4-24.5: 0
Frequency of numbers in range 24.5-24.6: 0
Frequency of numbers in range 24.6-24.7: 0
Frequency of numbers in range 24.7-24.8: 0
Frequency of numbers in range 24.8-24.9: 0
Frequency of numbers in range 24.9-25.0: 0
Frequency of numbers in range 25.0-25.1: 0
Frequency of numbers in range 25.1-25.2: 0
Frequency of numbers in range 25.2-25.3: 0
Frequency of numbers in range 25.3-25.4: 0
Frequency of numbers in range 25.4-25.5: 0
Frequency of numbers in range 25.5-25.6: 0
Frequency of numbers in range 25.6-25.7: 0
Frequency of numbers in range 25.7-25.8: 0
Frequency of numbers in range 25.8-25.9: 0
Frequency of numbers in range 25.9-26.0: 0
Frequency of numbers in range 26.0-26.1: 0
Frequency of numbers in range 26.1-26.2: 0
Frequency of numbers in range 26.2-26.3: 0
Frequency of numbers in range 26.3-26.4: 0
Frequency of numbers in range 26.4-26.5: 0
Frequency of numbers in range 26.5-26.6: 0
Frequency of numbers in range 26.6-26.7: 0
Frequency of numbers in range 26.7-26.8: 0
Frequency of numbers in range 26.8-26.9: 0
Frequency of numbers in range 26.9-27.0: 0
Frequency of numbers in range 27.0-27.1: 0
Frequency of numbers in range 27.1-27.2: 0
Frequency of numbers in range 27.2-27.3: 0
Frequency of numbers in range 27.3-27.4: 0
Frequency of numbers in range 27.4-27.5: 0
Frequency of numbers in range 27.5-27.6: 0
Frequency of numbers in range 27.6-27.7: 0
Frequency of numbers in range 27.7-27.8: 0
Frequency of numbers in range 27.8-27.9: 0
Frequency of numbers in range 27.9-28.0: 0
Frequency of numbers in range 28.0-28.1: 0
Frequency of numbers in range 28.1-28.2: 0
Frequency of numbers in range 28.2-28.3: 0
Frequency of numbers in range 28.3-28.4: 0
Frequency of numbers in range 28.4-28.5: 0
Frequency of numbers in range 28.5-28.6: 0
Frequency of numbers in range 28.6-28.7: 0
Frequency of numbers in range 28.7-28.8: 0
Frequency of numbers in range 28.8-28.9: 0
Frequency of numbers in range 28.9-29.0: 0
Frequency of numbers in range 29.0-29.1: 0
Frequency of numbers in range 29.1-29.2: 0
Frequency of numbers in range 29.2-29.3: 0
Frequency of numbers in range 29.3-29.4: 0
Frequency of numbers in range 29.4-29.5: 0
Frequency of numbers in range 29.5-29.6: 0

```
Frequency of numbers in range 29.6-29.7: 0
Frequency of numbers in range 29.7-29.8: 0
Frequency of numbers in range 29.8-29.9: 0
Frequency of numbers in range 29.9-30.0: 0
Frequency of numbers in range 30.0-30.1: 0
Frequency of numbers in range 30.1-30.2: 0
Frequency of numbers in range 30.2-30.3: 0
Frequency of numbers in range 30.3-30.4: 0
Frequency of numbers in range 30.4-30.5: 0
Frequency of numbers in range 30.5-30.6: 0
Frequency of numbers in range 30.6-30.7: 0
Frequency of numbers in range 30.7-30.8: 0
Frequency of numbers in range 30.8-30.9: 0
Frequency of numbers in range 30.9-31.0: 0
Frequency of numbers in range 31.0-31.1: 0
Frequency of numbers in range 31.1-31.2: 0
Frequency of numbers in range 31.2-31.3: 0
Frequency of numbers in range 31.3-31.4: 0
Frequency of numbers in range 31.4-31.5: 0
Frequency of numbers in range 31.5-31.6: 0
Frequency of numbers in range 31.6-31.7: 0
Frequency of numbers in range 31.7-31.8: 0
Frequency of numbers in range 31.8-31.9: 0
Frequency of numbers in range 31.9-32.0: 0
Frequency of numbers in range 32.0-32.1: 0
Frequency of numbers in range 32.1-32.2: 0
Frequency of numbers in range 32.2-32.3: 0
Frequency of numbers in range 32.3-32.4: 0
Frequency of numbers in range 32.4-32.5: 0
Frequency of numbers in range 32.5-32.6: 0
Frequency of numbers in range 32.6-32.7: 0
Frequency of numbers in range 32.7-32.8: 0
Frequency of numbers in range 32.8-32.9: 0
Frequency of numbers in range 32.9-33.0: 0
Frequency of numbers in range 33.0-33.1: 0
Frequency of numbers in range 33.1-33.2: 0
Frequency of numbers in range 33.2-33.3: 0
Frequency of numbers in range 33.3-33.4: 0
Frequency of numbers in range 33.4-33.5: 0
Frequency of numbers in range 33.5-33.6: 0
Frequency of numbers in range 33.6-33.7: 0
Frequency of numbers in range 33.7-33.8: 0
Frequency of numbers in range 33.8-33.9: 0
Frequency of numbers in range 33.9-34.0: 0
Frequency of numbers in range 34.0-34.1: 0
Frequency of numbers in range 34.1-34.2: 0
Frequency of numbers in range 34.2-34.3: 0
Frequency of numbers in range 34.3-34.4: 0
Frequency of numbers in range 34.4-34.5: 0
Frequency of numbers in range 34.5-34.6: 0
Frequency of numbers in range 34.6-34.7: 0
Frequency of numbers in range 34.7-34.8: 0
Frequency of numbers in range 34.8-34.9: 0
Frequency of numbers in range 34.9-35.0: 0
Frequency of numbers in range 35.0-35.1: 0
Frequency of numbers in range 35.1-35.2: 0
```

Frequency of numbers in range 35.2-35.3: 0
Frequency of numbers in range 35.3-35.4: 0
Frequency of numbers in range 35.4-35.5: 0
Frequency of numbers in range 35.5-35.6: 0
Frequency of numbers in range 35.6-35.7: 0
Frequency of numbers in range 35.7-35.8: 0
Frequency of numbers in range 35.8-35.9: 0
Frequency of numbers in range 35.9-36.0: 0
Frequency of numbers in range 36.0-36.1: 0
Frequency of numbers in range 36.1-36.2: 0
Frequency of numbers in range 36.2-36.3: 0
Frequency of numbers in range 36.3-36.4: 0
Frequency of numbers in range 36.4-36.5: 0
Frequency of numbers in range 36.5-36.6: 0
Frequency of numbers in range 36.6-36.7: 0
Frequency of numbers in range 36.7-36.8: 0
Frequency of numbers in range 36.8-36.9: 0
Frequency of numbers in range 36.9-37.0: 0
Frequency of numbers in range 37.0-37.1: 0
Frequency of numbers in range 37.1-37.2: 0
Frequency of numbers in range 37.2-37.3: 0
Frequency of numbers in range 37.3-37.4: 0
Frequency of numbers in range 37.4-37.5: 0
Frequency of numbers in range 37.5-37.6: 0
Frequency of numbers in range 37.6-37.7: 0
Frequency of numbers in range 37.7-37.8: 0
Frequency of numbers in range 37.8-37.9: 0
Frequency of numbers in range 37.9-38.0: 0
Frequency of numbers in range 38.0-38.1: 0
Frequency of numbers in range 38.1-38.2: 0
Frequency of numbers in range 38.2-38.3: 0
Frequency of numbers in range 38.3-38.4: 0
Frequency of numbers in range 38.4-38.5: 0
Frequency of numbers in range 38.5-38.6: 0
Frequency of numbers in range 38.6-38.7: 0
Frequency of numbers in range 38.7-38.8: 0
Frequency of numbers in range 38.8-38.9: 0
Frequency of numbers in range 38.9-39.0: 0
Frequency of numbers in range 39.0-39.1: 0
Frequency of numbers in range 39.1-39.2: 0
Frequency of numbers in range 39.2-39.3: 0
Frequency of numbers in range 39.3-39.4: 0
Frequency of numbers in range 39.4-39.5: 0
Frequency of numbers in range 39.5-39.6: 0
Frequency of numbers in range 39.6-39.7: 0
Frequency of numbers in range 39.7-39.8: 0
Frequency of numbers in range 39.8-39.9: 0
Frequency of numbers in range 39.9-40.0: 0
Frequency of numbers in range 40.0-40.1: 0
Frequency of numbers in range 40.1-40.2: 0
Frequency of numbers in range 40.2-40.3: 0
Frequency of numbers in range 40.3-40.4: 0
Frequency of numbers in range 40.4-40.5: 0
Frequency of numbers in range 40.5-40.6: 0
Frequency of numbers in range 40.6-40.7: 0
Frequency of numbers in range 40.7-40.8: 0

Frequency of numbers in range 40.8-40.9: 0
Frequency of numbers in range 40.9-41.0: 0
Frequency of numbers in range 41.0-41.1: 0
Frequency of numbers in range 41.1-41.2: 0
Frequency of numbers in range 41.2-41.3: 0
Frequency of numbers in range 41.3-41.4: 0
Frequency of numbers in range 41.4-41.5: 0
Frequency of numbers in range 41.5-41.6: 0
Frequency of numbers in range 41.6-41.7: 0
Frequency of numbers in range 41.7-41.8: 0
Frequency of numbers in range 41.8-41.9: 0
Frequency of numbers in range 41.9-42.0: 0
Frequency of numbers in range 42.0-42.1: 0
Frequency of numbers in range 42.1-42.2: 0
Frequency of numbers in range 42.2-42.3: 0
Frequency of numbers in range 42.3-42.4: 0
Frequency of numbers in range 42.4-42.5: 0
Frequency of numbers in range 42.5-42.6: 0
Frequency of numbers in range 42.6-42.7: 0
Frequency of numbers in range 42.7-42.8: 0
Frequency of numbers in range 42.8-42.9: 0
Frequency of numbers in range 42.9-43.0: 0
Frequency of numbers in range 43.0-43.1: 0
Frequency of numbers in range 43.1-43.2: 0
Frequency of numbers in range 43.2-43.3: 0
Frequency of numbers in range 43.3-43.4: 0
Frequency of numbers in range 43.4-43.5: 0
Frequency of numbers in range 43.5-43.6: 0
Frequency of numbers in range 43.6-43.7: 0
Frequency of numbers in range 43.7-43.8: 0
Frequency of numbers in range 43.8-43.9: 0
Frequency of numbers in range 43.9-44.0: 0
Frequency of numbers in range 44.0-44.1: 0
Frequency of numbers in range 44.1-44.2: 0
Frequency of numbers in range 44.2-44.3: 0
Frequency of numbers in range 44.3-44.4: 0
Frequency of numbers in range 44.4-44.5: 0
Frequency of numbers in range 44.5-44.6: 0
Frequency of numbers in range 44.6-44.7: 0
Frequency of numbers in range 44.7-44.8: 0
Frequency of numbers in range 44.8-44.9: 0
Frequency of numbers in range 44.9-45.0: 0
Frequency of numbers in range 45.0-45.1: 0
Frequency of numbers in range 45.1-45.2: 0
Frequency of numbers in range 45.2-45.3: 0
Frequency of numbers in range 45.3-45.4: 0
Frequency of numbers in range 45.4-45.5: 0
Frequency of numbers in range 45.5-45.6: 0
Frequency of numbers in range 45.6-45.7: 0
Frequency of numbers in range 45.7-45.8: 0
Frequency of numbers in range 45.8-45.9: 0
Frequency of numbers in range 45.9-46.0: 0
Frequency of numbers in range 46.0-46.1: 0
Frequency of numbers in range 46.1-46.2: 0
Frequency of numbers in range 46.2-46.3: 0
Frequency of numbers in range 46.3-46.4: 0

```
Frequency of numbers in range 46.4-46.5: 0
Frequency of numbers in range 46.5-46.6: 0
Frequency of numbers in range 46.6-46.7: 0
Frequency of numbers in range 46.7-46.8: 0
Frequency of numbers in range 46.8-46.9: 0
Frequency of numbers in range 46.9-47.0: 0
Frequency of numbers in range 47.0-47.1: 0
Frequency of numbers in range 47.1-47.2: 0
Frequency of numbers in range 47.2-47.3: 0
Frequency of numbers in range 47.3-47.4: 0
Frequency of numbers in range 47.4-47.5: 0
Frequency of numbers in range 47.5-47.6: 0
Frequency of numbers in range 47.6-47.7: 0
Frequency of numbers in range 47.7-47.8: 0
Frequency of numbers in range 47.8-47.9: 0
Frequency of numbers in range 47.9-48.0: 0
Frequency of numbers in range 48.0-48.1: 0
Frequency of numbers in range 48.1-48.2: 0
Frequency of numbers in range 48.2-48.3: 0
Frequency of numbers in range 48.3-48.4: 0
Frequency of numbers in range 48.4-48.5: 0
Frequency of numbers in range 48.5-48.6: 0
Frequency of numbers in range 48.6-48.7: 0
Frequency of numbers in range 48.7-48.8: 0
Frequency of numbers in range 48.8-48.9: 0
Frequency of numbers in range 48.9-49.0: 0
Frequency of numbers in range 49.0-49.1: 0
Frequency of numbers in range 49.1-49.2: 0
Frequency of numbers in range 49.2-49.3: 0
Frequency of numbers in range 49.3-49.4: 0
Frequency of numbers in range 49.4-49.5: 0
Frequency of numbers in range 49.5-49.6: 0
Frequency of numbers in range 49.6-49.7: 0
Frequency of numbers in range 49.7-49.8: 0
Frequency of numbers in range 49.8-49.9: 0
Frequency of numbers in range 49.9-50.0: 0
Frequency of numbers in range 50.0-50.1: 0
Frequency of numbers in range 50.1-50.2: 0
Frequency of numbers in range 50.2-50.3: 0
Frequency of numbers in range 50.3-50.4: 0
Frequency of numbers in range 50.4-50.5: 0
Frequency of numbers in range 50.5-50.6: 0
Frequency of numbers in range 50.6-50.7: 0
Frequency of numbers in range 50.7-50.8: 0
Frequency of numbers in range 50.8-50.9: 0
Frequency of numbers in range 50.9-51.0: 0
Frequency of numbers in range 51.0-51.1: 0
Frequency of numbers in range 51.1-51.2: 0
Frequency of numbers in range 51.2-51.3: 0
Frequency of numbers in range 51.3-51.4: 0
Frequency of numbers in range 51.4-51.5: 0
Frequency of numbers in range 51.5-51.6: 0
Frequency of numbers in range 51.6-51.7: 0
Frequency of numbers in range 51.7-51.8: 0
Frequency of numbers in range 51.8-51.9: 0
Frequency of numbers in range 51.9-52.0: 0
```

Frequency of numbers in range 52.0-52.1: 0
Frequency of numbers in range 52.1-52.2: 0
Frequency of numbers in range 52.2-52.3: 0
Frequency of numbers in range 52.3-52.4: 0
Frequency of numbers in range 52.4-52.5: 0
Frequency of numbers in range 52.5-52.6: 0
Frequency of numbers in range 52.6-52.7: 0
Frequency of numbers in range 52.7-52.8: 0
Frequency of numbers in range 52.8-52.9: 0
Frequency of numbers in range 52.9-53.0: 0
Frequency of numbers in range 53.0-53.1: 0
Frequency of numbers in range 53.1-53.2: 0
Frequency of numbers in range 53.2-53.3: 0
Frequency of numbers in range 53.3-53.4: 0
Frequency of numbers in range 53.4-53.5: 0
Frequency of numbers in range 53.5-53.6: 0
Frequency of numbers in range 53.6-53.7: 0
Frequency of numbers in range 53.7-53.8: 0
Frequency of numbers in range 53.8-53.9: 0
Frequency of numbers in range 53.9-54.0: 0
Frequency of numbers in range 54.0-54.1: 0
Frequency of numbers in range 54.1-54.2: 0
Frequency of numbers in range 54.2-54.3: 0
Frequency of numbers in range 54.3-54.4: 0
Frequency of numbers in range 54.4-54.5: 0
Frequency of numbers in range 54.5-54.6: 0
Frequency of numbers in range 54.6-54.7: 0
Frequency of numbers in range 54.7-54.8: 0
Frequency of numbers in range 54.8-54.9: 0
Frequency of numbers in range 54.9-55.0: 0
Frequency of numbers in range 55.0-55.1: 0
Frequency of numbers in range 55.1-55.2: 0
Frequency of numbers in range 55.2-55.3: 0
Frequency of numbers in range 55.3-55.4: 0
Frequency of numbers in range 55.4-55.5: 0
Frequency of numbers in range 55.5-55.6: 0
Frequency of numbers in range 55.6-55.7: 0
Frequency of numbers in range 55.7-55.8: 0
Frequency of numbers in range 55.8-55.9: 0
Frequency of numbers in range 55.9-56.0: 0
Frequency of numbers in range 56.0-56.1: 0
Frequency of numbers in range 56.1-56.2: 0
Frequency of numbers in range 56.2-56.3: 0
Frequency of numbers in range 56.3-56.4: 0
Frequency of numbers in range 56.4-56.5: 0
Frequency of numbers in range 56.5-56.6: 0
Frequency of numbers in range 56.6-56.7: 0
Frequency of numbers in range 56.7-56.8: 0
Frequency of numbers in range 56.8-56.9: 0
Frequency of numbers in range 56.9-57.0: 0
Frequency of numbers in range 57.0-57.1: 0
Frequency of numbers in range 57.1-57.2: 0
Frequency of numbers in range 57.2-57.3: 0
Frequency of numbers in range 57.3-57.4: 0
Frequency of numbers in range 57.4-57.5: 0
Frequency of numbers in range 57.5-57.6: 0

```
Frequency of numbers in range 57.6-57.7: 0
Frequency of numbers in range 57.7-57.8: 0
Frequency of numbers in range 57.8-57.9: 0
Frequency of numbers in range 57.9-58.0: 0
Frequency of numbers in range 58.0-58.1: 0
Frequency of numbers in range 58.1-58.2: 0
Frequency of numbers in range 58.2-58.3: 0
Frequency of numbers in range 58.3-58.4: 0
Frequency of numbers in range 58.4-58.5: 0
Frequency of numbers in range 58.5-58.6: 0
Frequency of numbers in range 58.6-58.7: 0
Frequency of numbers in range 58.7-58.8: 0
Frequency of numbers in range 58.8-58.9: 0
Frequency of numbers in range 58.9-59.0: 0
Frequency of numbers in range 59.0-59.1: 0
Frequency of numbers in range 59.1-59.2: 0
Frequency of numbers in range 59.2-59.3: 0
Frequency of numbers in range 59.3-59.4: 0
Frequency of numbers in range 59.4-59.5: 0
Frequency of numbers in range 59.5-59.6: 0
Frequency of numbers in range 59.6-59.7: 0
Frequency of numbers in range 59.7-59.8: 0
Frequency of numbers in range 59.8-59.9: 0
Frequency of numbers in range 59.9-60.0: 0
Frequency of numbers in range 60.0-60.1: 0
Frequency of numbers in range 60.1-60.2: 0
Frequency of numbers in range 60.2-60.3: 0
Frequency of numbers in range 60.3-60.4: 0
Frequency of numbers in range 60.4-60.5: 0
Frequency of numbers in range 60.5-60.6: 0
Frequency of numbers in range 60.6-60.7: 0
Frequency of numbers in range 60.7-60.8: 0
Frequency of numbers in range 60.8-60.9: 0
Frequency of numbers in range 60.9-61.0: 0
Frequency of numbers in range 61.0-61.1: 0
Frequency of numbers in range 61.1-61.2: 0
Frequency of numbers in range 61.2-61.3: 0
Frequency of numbers in range 61.3-61.4: 0
Frequency of numbers in range 61.4-61.5: 0
Frequency of numbers in range 61.5-61.6: 0
Frequency of numbers in range 61.6-61.7: 0
Frequency of numbers in range 61.7-61.8: 0
Frequency of numbers in range 61.8-61.9: 0
Frequency of numbers in range 61.9-62.0: 0
Frequency of numbers in range 62.0-62.1: 0
Frequency of numbers in range 62.1-62.2: 0
Frequency of numbers in range 62.2-62.3: 0
Frequency of numbers in range 62.3-62.4: 0
Frequency of numbers in range 62.4-62.5: 0
Frequency of numbers in range 62.5-62.6: 0
Frequency of numbers in range 62.6-62.7: 0
Frequency of numbers in range 62.7-62.8: 0
Frequency of numbers in range 62.8-62.9: 0
Frequency of numbers in range 62.9-63.0: 0
Frequency of numbers in range 63.0-63.1: 0
Frequency of numbers in range 63.1-63.2: 0
```

Frequency of numbers in range 63.2-63.3: 0
Frequency of numbers in range 63.3-63.4: 0
Frequency of numbers in range 63.4-63.5: 0
Frequency of numbers in range 63.5-63.6: 0
Frequency of numbers in range 63.6-63.7: 0
Frequency of numbers in range 63.7-63.8: 0
Frequency of numbers in range 63.8-63.9: 0
Frequency of numbers in range 63.9-64.0: 0
Frequency of numbers in range 64.0-64.1: 0
Frequency of numbers in range 64.1-64.2: 0
Frequency of numbers in range 64.2-64.3: 0
Frequency of numbers in range 64.3-64.4: 0
Frequency of numbers in range 64.4-64.5: 0
Frequency of numbers in range 64.5-64.6: 0
Frequency of numbers in range 64.6-64.7: 0
Frequency of numbers in range 64.7-64.8: 0
Frequency of numbers in range 64.8-64.9: 0
Frequency of numbers in range 64.9-65.0: 0
Frequency of numbers in range 65.0-65.1: 0
Frequency of numbers in range 65.1-65.2: 0
Frequency of numbers in range 65.2-65.3: 0
Frequency of numbers in range 65.3-65.4: 0
Frequency of numbers in range 65.4-65.5: 0
Frequency of numbers in range 65.5-65.6: 0
Frequency of numbers in range 65.6-65.7: 0
Frequency of numbers in range 65.7-65.8: 0
Frequency of numbers in range 65.8-65.9: 0
Frequency of numbers in range 65.9-66.0: 0
Frequency of numbers in range 66.0-66.1: 0
Frequency of numbers in range 66.1-66.2: 0
Frequency of numbers in range 66.2-66.3: 0
Frequency of numbers in range 66.3-66.4: 0
Frequency of numbers in range 66.4-66.5: 0
Frequency of numbers in range 66.5-66.6: 0
Frequency of numbers in range 66.6-66.7: 0
Frequency of numbers in range 66.7-66.8: 0
Frequency of numbers in range 66.8-66.9: 0
Frequency of numbers in range 66.9-67.0: 0
Frequency of numbers in range 67.0-67.1: 0
Frequency of numbers in range 67.1-67.2: 0
Frequency of numbers in range 67.2-67.3: 0
Frequency of numbers in range 67.3-67.4: 0
Frequency of numbers in range 67.4-67.5: 0
Frequency of numbers in range 67.5-67.6: 0
Frequency of numbers in range 67.6-67.7: 0
Frequency of numbers in range 67.7-67.8: 0
Frequency of numbers in range 67.8-67.9: 0
Frequency of numbers in range 67.9-68.0: 0
Frequency of numbers in range 68.0-68.1: 0
Frequency of numbers in range 68.1-68.2: 0
Frequency of numbers in range 68.2-68.3: 0
Frequency of numbers in range 68.3-68.4: 0
Frequency of numbers in range 68.4-68.5: 0
Frequency of numbers in range 68.5-68.6: 0
Frequency of numbers in range 68.6-68.7: 0
Frequency of numbers in range 68.7-68.8: 0

Frequency of numbers in range 68.8-68.9: 0
Frequency of numbers in range 68.9-69.0: 0
Frequency of numbers in range 69.0-69.1: 0
Frequency of numbers in range 69.1-69.2: 0
Frequency of numbers in range 69.2-69.3: 0
Frequency of numbers in range 69.3-69.4: 0
Frequency of numbers in range 69.4-69.5: 0
Frequency of numbers in range 69.5-69.6: 0
Frequency of numbers in range 69.6-69.7: 0
Frequency of numbers in range 69.7-69.8: 0
Frequency of numbers in range 69.8-69.9: 0
Frequency of numbers in range 69.9-70.0: 0
Frequency of numbers in range 70.0-70.1: 0
Frequency of numbers in range 70.1-70.2: 0
Frequency of numbers in range 70.2-70.3: 0
Frequency of numbers in range 70.3-70.4: 0
Frequency of numbers in range 70.4-70.5: 0
Frequency of numbers in range 70.5-70.6: 0
Frequency of numbers in range 70.6-70.7: 0
Frequency of numbers in range 70.7-70.8: 0
Frequency of numbers in range 70.8-70.9: 0
Frequency of numbers in range 70.9-71.0: 0
Frequency of numbers in range 71.0-71.1: 0
Frequency of numbers in range 71.1-71.2: 0
Frequency of numbers in range 71.2-71.3: 0
Frequency of numbers in range 71.3-71.4: 0
Frequency of numbers in range 71.4-71.5: 0
Frequency of numbers in range 71.5-71.6: 0
Frequency of numbers in range 71.6-71.7: 0
Frequency of numbers in range 71.7-71.8: 0
Frequency of numbers in range 71.8-71.9: 0
Frequency of numbers in range 71.9-72.0: 0
Frequency of numbers in range 72.0-72.1: 0
Frequency of numbers in range 72.1-72.2: 0
Frequency of numbers in range 72.2-72.3: 0
Frequency of numbers in range 72.3-72.4: 0
Frequency of numbers in range 72.4-72.5: 0
Frequency of numbers in range 72.5-72.6: 0
Frequency of numbers in range 72.6-72.7: 0
Frequency of numbers in range 72.7-72.8: 0
Frequency of numbers in range 72.8-72.9: 0
Frequency of numbers in range 72.9-73.0: 0
Frequency of numbers in range 73.0-73.1: 0
Frequency of numbers in range 73.1-73.2: 0
Frequency of numbers in range 73.2-73.3: 0
Frequency of numbers in range 73.3-73.4: 0
Frequency of numbers in range 73.4-73.5: 0
Frequency of numbers in range 73.5-73.6: 0
Frequency of numbers in range 73.6-73.7: 0
Frequency of numbers in range 73.7-73.8: 0
Frequency of numbers in range 73.8-73.9: 0
Frequency of numbers in range 73.9-74.0: 0
Frequency of numbers in range 74.0-74.1: 0
Frequency of numbers in range 74.1-74.2: 0
Frequency of numbers in range 74.2-74.3: 0
Frequency of numbers in range 74.3-74.4: 0

Frequency of numbers in range 74.4-74.5: 0
Frequency of numbers in range 74.5-74.6: 0
Frequency of numbers in range 74.6-74.7: 0
Frequency of numbers in range 74.7-74.8: 0
Frequency of numbers in range 74.8-74.9: 0
Frequency of numbers in range 74.9-75.0: 0
Frequency of numbers in range 75.0-75.1: 0
Frequency of numbers in range 75.1-75.2: 0
Frequency of numbers in range 75.2-75.3: 0
Frequency of numbers in range 75.3-75.4: 0
Frequency of numbers in range 75.4-75.5: 0
Frequency of numbers in range 75.5-75.6: 0
Frequency of numbers in range 75.6-75.7: 0
Frequency of numbers in range 75.7-75.8: 0
Frequency of numbers in range 75.8-75.9: 0
Frequency of numbers in range 75.9-76.0: 0
Frequency of numbers in range 76.0-76.1: 0
Frequency of numbers in range 76.1-76.2: 0
Frequency of numbers in range 76.2-76.3: 0
Frequency of numbers in range 76.3-76.4: 0
Frequency of numbers in range 76.4-76.5: 0
Frequency of numbers in range 76.5-76.6: 0
Frequency of numbers in range 76.6-76.7: 0
Frequency of numbers in range 76.7-76.8: 0
Frequency of numbers in range 76.8-76.9: 0
Frequency of numbers in range 76.9-77.0: 0
Frequency of numbers in range 77.0-77.1: 0
Frequency of numbers in range 77.1-77.2: 0
Frequency of numbers in range 77.2-77.3: 0
Frequency of numbers in range 77.3-77.4: 0
Frequency of numbers in range 77.4-77.5: 0
Frequency of numbers in range 77.5-77.6: 0
Frequency of numbers in range 77.6-77.7: 0
Frequency of numbers in range 77.7-77.8: 0
Frequency of numbers in range 77.8-77.9: 0
Frequency of numbers in range 77.9-78.0: 0
Frequency of numbers in range 78.0-78.1: 0
Frequency of numbers in range 78.1-78.2: 0
Frequency of numbers in range 78.2-78.3: 0
Frequency of numbers in range 78.3-78.4: 0
Frequency of numbers in range 78.4-78.5: 0
Frequency of numbers in range 78.5-78.6: 0
Frequency of numbers in range 78.6-78.7: 0
Frequency of numbers in range 78.7-78.8: 0
Frequency of numbers in range 78.8-78.9: 0
Frequency of numbers in range 78.9-79.0: 0
Frequency of numbers in range 79.0-79.1: 0
Frequency of numbers in range 79.1-79.2: 0
Frequency of numbers in range 79.2-79.3: 0
Frequency of numbers in range 79.3-79.4: 0
Frequency of numbers in range 79.4-79.5: 0
Frequency of numbers in range 79.5-79.6: 0
Frequency of numbers in range 79.6-79.7: 0
Frequency of numbers in range 79.7-79.8: 0
Frequency of numbers in range 79.8-79.9: 0
Frequency of numbers in range 79.9-80.0: 0

Frequency of numbers in range 80.0-80.1: 0
Frequency of numbers in range 80.1-80.2: 0
Frequency of numbers in range 80.2-80.3: 0
Frequency of numbers in range 80.3-80.4: 0
Frequency of numbers in range 80.4-80.5: 0
Frequency of numbers in range 80.5-80.6: 0
Frequency of numbers in range 80.6-80.7: 0
Frequency of numbers in range 80.7-80.8: 0
Frequency of numbers in range 80.8-80.9: 0
Frequency of numbers in range 80.9-81.0: 0
Frequency of numbers in range 81.0-81.1: 0
Frequency of numbers in range 81.1-81.2: 0
Frequency of numbers in range 81.2-81.3: 0
Frequency of numbers in range 81.3-81.4: 0
Frequency of numbers in range 81.4-81.5: 0
Frequency of numbers in range 81.5-81.6: 0
Frequency of numbers in range 81.6-81.7: 0
Frequency of numbers in range 81.7-81.8: 0
Frequency of numbers in range 81.8-81.9: 0
Frequency of numbers in range 81.9-82.0: 0
Frequency of numbers in range 82.0-82.1: 0
Frequency of numbers in range 82.1-82.2: 0
Frequency of numbers in range 82.2-82.3: 0
Frequency of numbers in range 82.3-82.4: 0
Frequency of numbers in range 82.4-82.5: 0
Frequency of numbers in range 82.5-82.6: 0
Frequency of numbers in range 82.6-82.7: 0
Frequency of numbers in range 82.7-82.8: 0
Frequency of numbers in range 82.8-82.9: 0
Frequency of numbers in range 82.9-83.0: 0
Frequency of numbers in range 83.0-83.1: 0
Frequency of numbers in range 83.1-83.2: 0
Frequency of numbers in range 83.2-83.3: 0
Frequency of numbers in range 83.3-83.4: 0
Frequency of numbers in range 83.4-83.5: 0
Frequency of numbers in range 83.5-83.6: 0
Frequency of numbers in range 83.6-83.7: 0
Frequency of numbers in range 83.7-83.8: 0
Frequency of numbers in range 83.8-83.9: 0
Frequency of numbers in range 83.9-84.0: 0
Frequency of numbers in range 84.0-84.1: 0
Frequency of numbers in range 84.1-84.2: 0
Frequency of numbers in range 84.2-84.3: 0
Frequency of numbers in range 84.3-84.4: 0
Frequency of numbers in range 84.4-84.5: 0
Frequency of numbers in range 84.5-84.6: 0
Frequency of numbers in range 84.6-84.7: 0
Frequency of numbers in range 84.7-84.8: 0
Frequency of numbers in range 84.8-84.9: 0
Frequency of numbers in range 84.9-85.0: 0
Frequency of numbers in range 85.0-85.1: 0
Frequency of numbers in range 85.1-85.2: 0
Frequency of numbers in range 85.2-85.3: 0
Frequency of numbers in range 85.3-85.4: 0
Frequency of numbers in range 85.4-85.5: 0
Frequency of numbers in range 85.5-85.6: 0

Frequency of numbers in range 85.6-85.7: 0
Frequency of numbers in range 85.7-85.8: 0
Frequency of numbers in range 85.8-85.9: 0
Frequency of numbers in range 85.9-86.0: 0
Frequency of numbers in range 86.0-86.1: 0
Frequency of numbers in range 86.1-86.2: 0
Frequency of numbers in range 86.2-86.3: 0
Frequency of numbers in range 86.3-86.4: 0
Frequency of numbers in range 86.4-86.5: 0
Frequency of numbers in range 86.5-86.6: 0
Frequency of numbers in range 86.6-86.7: 0
Frequency of numbers in range 86.7-86.8: 0
Frequency of numbers in range 86.8-86.9: 0
Frequency of numbers in range 86.9-87.0: 0
Frequency of numbers in range 87.0-87.1: 0
Frequency of numbers in range 87.1-87.2: 0
Frequency of numbers in range 87.2-87.3: 0
Frequency of numbers in range 87.3-87.4: 0
Frequency of numbers in range 87.4-87.5: 0
Frequency of numbers in range 87.5-87.6: 0
Frequency of numbers in range 87.6-87.7: 0
Frequency of numbers in range 87.7-87.8: 0
Frequency of numbers in range 87.8-87.9: 0
Frequency of numbers in range 87.9-88.0: 0
Frequency of numbers in range 88.0-88.1: 0
Frequency of numbers in range 88.1-88.2: 0
Frequency of numbers in range 88.2-88.3: 0
Frequency of numbers in range 88.3-88.4: 0
Frequency of numbers in range 88.4-88.5: 0
Frequency of numbers in range 88.5-88.6: 0
Frequency of numbers in range 88.6-88.7: 0
Frequency of numbers in range 88.7-88.8: 0
Frequency of numbers in range 88.8-88.9: 0
Frequency of numbers in range 88.9-89.0: 0
Frequency of numbers in range 89.0-89.1: 0
Frequency of numbers in range 89.1-89.2: 0
Frequency of numbers in range 89.2-89.3: 0
Frequency of numbers in range 89.3-89.4: 0
Frequency of numbers in range 89.4-89.5: 0
Frequency of numbers in range 89.5-89.6: 0
Frequency of numbers in range 89.6-89.7: 0
Frequency of numbers in range 89.7-89.8: 0
Frequency of numbers in range 89.8-89.9: 0
Frequency of numbers in range 89.9-90.0: 0
Frequency of numbers in range 90.0-90.1: 0
Frequency of numbers in range 90.1-90.2: 0
Frequency of numbers in range 90.2-90.3: 0
Frequency of numbers in range 90.3-90.4: 0
Frequency of numbers in range 90.4-90.5: 0
Frequency of numbers in range 90.5-90.6: 0
Frequency of numbers in range 90.6-90.7: 0
Frequency of numbers in range 90.7-90.8: 0
Frequency of numbers in range 90.8-90.9: 0
Frequency of numbers in range 90.9-91.0: 0
Frequency of numbers in range 91.0-91.1: 0
Frequency of numbers in range 91.1-91.2: 0

Frequency of numbers in range 91.2-91.3: 0
Frequency of numbers in range 91.3-91.4: 0
Frequency of numbers in range 91.4-91.5: 0
Frequency of numbers in range 91.5-91.6: 0
Frequency of numbers in range 91.6-91.7: 0
Frequency of numbers in range 91.7-91.8: 0
Frequency of numbers in range 91.8-91.9: 0
Frequency of numbers in range 91.9-92.0: 0
Frequency of numbers in range 92.0-92.1: 0
Frequency of numbers in range 92.1-92.2: 0
Frequency of numbers in range 92.2-92.3: 0
Frequency of numbers in range 92.3-92.4: 0
Frequency of numbers in range 92.4-92.5: 0
Frequency of numbers in range 92.5-92.6: 0
Frequency of numbers in range 92.6-92.7: 0
Frequency of numbers in range 92.7-92.8: 0
Frequency of numbers in range 92.8-92.9: 0
Frequency of numbers in range 92.9-93.0: 0
Frequency of numbers in range 93.0-93.1: 0
Frequency of numbers in range 93.1-93.2: 0
Frequency of numbers in range 93.2-93.3: 0
Frequency of numbers in range 93.3-93.4: 0
Frequency of numbers in range 93.4-93.5: 0
Frequency of numbers in range 93.5-93.6: 0
Frequency of numbers in range 93.6-93.7: 0
Frequency of numbers in range 93.7-93.8: 0
Frequency of numbers in range 93.8-93.9: 0
Frequency of numbers in range 93.9-94.0: 0
Frequency of numbers in range 94.0-94.1: 0
Frequency of numbers in range 94.1-94.2: 0
Frequency of numbers in range 94.2-94.3: 0
Frequency of numbers in range 94.3-94.4: 0
Frequency of numbers in range 94.4-94.5: 0
Frequency of numbers in range 94.5-94.6: 0
Frequency of numbers in range 94.6-94.7: 0
Frequency of numbers in range 94.7-94.8: 0
Frequency of numbers in range 94.8-94.9: 0
Frequency of numbers in range 94.9-95.0: 0
Frequency of numbers in range 95.0-95.1: 0
Frequency of numbers in range 95.1-95.2: 0
Frequency of numbers in range 95.2-95.3: 0
Frequency of numbers in range 95.3-95.4: 0
Frequency of numbers in range 95.4-95.5: 0
Frequency of numbers in range 95.5-95.6: 0
Frequency of numbers in range 95.6-95.7: 0
Frequency of numbers in range 95.7-95.8: 0
Frequency of numbers in range 95.8-95.9: 0
Frequency of numbers in range 95.9-96.0: 0
Frequency of numbers in range 96.0-96.1: 0
Frequency of numbers in range 96.1-96.2: 0
Frequency of numbers in range 96.2-96.3: 0
Frequency of numbers in range 96.3-96.4: 0
Frequency of numbers in range 96.4-96.5: 0
Frequency of numbers in range 96.5-96.6: 0
Frequency of numbers in range 96.6-96.7: 0
Frequency of numbers in range 96.7-96.8: 0

```
Frequency of numbers in range 96.8-96.9: 0
Frequency of numbers in range 96.9-97.0: 0
Frequency of numbers in range 97.0-97.1: 0
Frequency of numbers in range 97.1-97.2: 0
Frequency of numbers in range 97.2-97.3: 0
Frequency of numbers in range 97.3-97.4: 0
Frequency of numbers in range 97.4-97.5: 0
Frequency of numbers in range 97.5-97.6: 0
Frequency of numbers in range 97.6-97.7: 0
Frequency of numbers in range 97.7-97.8: 0
Frequency of numbers in range 97.8-97.9: 0
Frequency of numbers in range 97.9-98.0: 0
Frequency of numbers in range 98.0-98.1: 0
Frequency of numbers in range 98.1-98.2: 0
Frequency of numbers in range 98.2-98.3: 0
Frequency of numbers in range 98.3-98.4: 0
Frequency of numbers in range 98.4-98.5: 0
Frequency of numbers in range 98.5-98.6: 0
Frequency of numbers in range 98.6-98.7: 0
Frequency of numbers in range 98.7-98.8: 0
Frequency of numbers in range 98.8-98.9: 0
Frequency of numbers in range 98.9-99.0: 0
Frequency of numbers in range 99.0-99.1: 0
Frequency of numbers in range 99.1-99.2: 0
Frequency of numbers in range 99.2-99.3: 0
Frequency of numbers in range 99.3-99.4: 0
Frequency of numbers in range 99.4-99.5: 0
Frequency of numbers in range 99.5-99.6: 0
Frequency of numbers in range 99.6-99.7: 0
Frequency of numbers in range 99.7-99.8: 0
Frequency of numbers in range 99.8-99.9: 0
Frequency of numbers in range 99.9-100.0: 0
```

Observation

Since the target "ViolentCrimesPerPop" has continuous values, we need to discretize the values to create different classes. In order to discretize the values, we are trying to observe the frequency of numbers present in each range.

In [9]:
```
y_data=[]
for i in y:
    if i>=0.0 and i<0.1:
        y_data.append(1)
    elif i>= 0.1 and i<0.2:
        y_data.append(2)
    else:
        y_data.append(3)
```

In [10]:
```
dataset_dataframe['label'] = y_data
dataset_dataframe.head()
```

Out[10]:

	population	householdsize	racepctblack	racePctWhite	racePctAsian	racePctHisp	ageP

0	0.19	0.33	0.02	0.90	0.12	0.17
1	0.00	0.16	0.12	0.74	0.45	0.07
2	0.00	0.42	0.49	0.56	0.17	0.04
3	0.04	0.77	1.00	0.08	0.12	0.10
4	0.01	0.55	0.02	0.95	0.09	0.05

5 rows × 102 columns

Observation

After Observing the frequency of values present in each range, we have divided them into three classes.
1) The first class i.e Label = 1 contains values between 0.0 and 0.1.
2) The second class i.e Label = 2 contains values between 0.1 and 0.2.
3) The other values belong to class with Label = 3.

In [11]:
```
dataset_dataframe.describe().T
```

Out[11]:

	count	mean	std	min	25%	50%	75%	max
population	1994.0	0.057593	0.126906	0.0	0.01	0.02	0.05	1.0
householdsize	1994.0	0.463395	0.163717	0.0	0.35	0.44	0.54	1.0
racepctblack	1994.0	0.179629	0.253442	0.0	0.02	0.06	0.23	1.0
racePctWhite	1994.0	0.753716	0.244039	0.0	0.63	0.85	0.94	1.0
racePctAsian	1994.0	0.153681	0.208877	0.0	0.04	0.07	0.17	1.0
...	...	...	...	...	...	...	...	...
PopDens	1994.0	0.232854	0.203092	0.0	0.10	0.17	0.28	1.0
PctUsePubTrans	1994.0	0.161685	0.229055	0.0	0.02	0.07	0.19	1.0
LemasPctOfficDrugUn	1994.0	0.094052	0.240328	0.0	0.00	0.00	0.00	1.0
ViolentCrimesPerPop	1994.0	0.237979	0.232985	0.0	0.07	0.15	0.33	1.0
label	1994.0	2.083250	0.870484	1.0	1.00	2.00	3.00	3.0

102 rows × 8 columns

Observation

The above contains the statistical measures of various features.

In [12]:
```
dataset_dataframe.isna().sum()
```

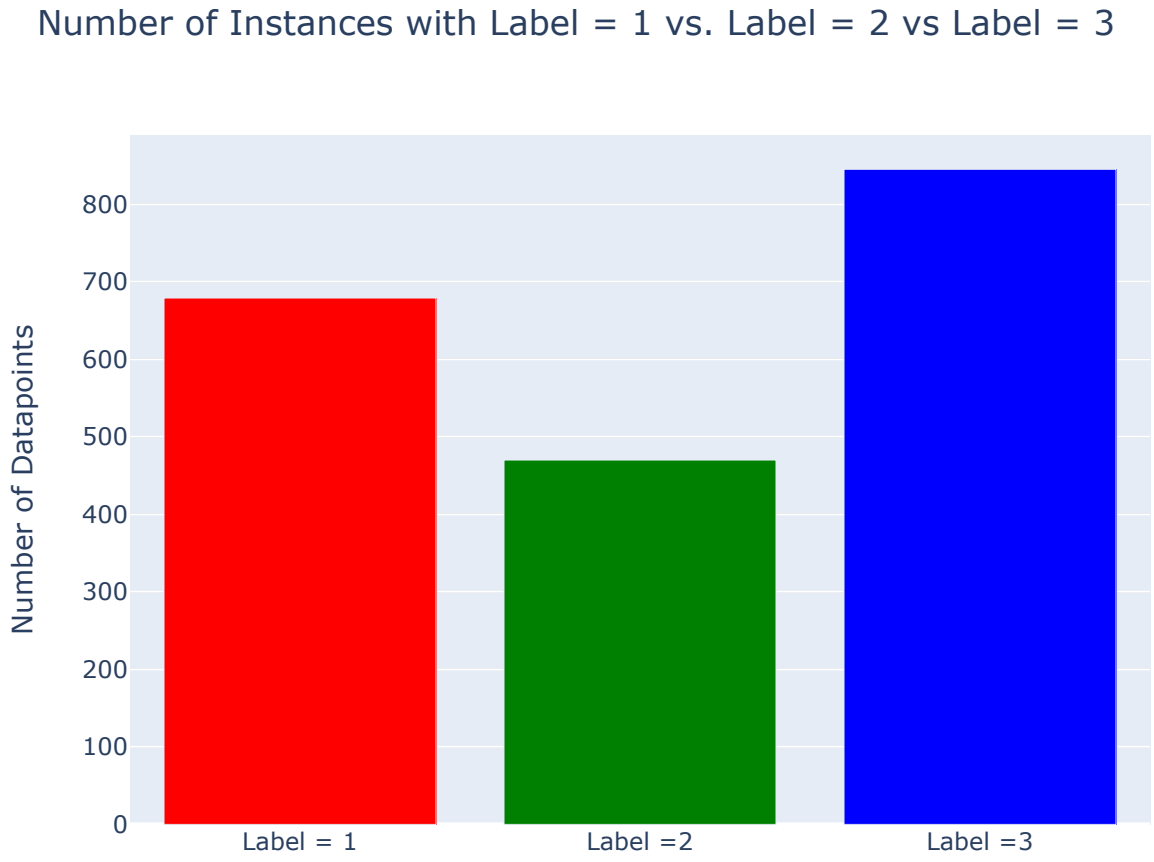
Out[12]:
```
population            0
```

```
householdsize          0
racepctblack           0
racePctWhite           0
racePctAsian           0
                      ..
PopDens                0
PctUsePubTrans         0
LemasPctOfficDrugUn    0
ViolentCrimesPerPop    0
label                  0
Length: 102, dtype: int64
```

Observation

There are no missing values in none of the features.

In [13]:
```
num_datapoints_label_1 = (dataset_dataframe['label'] == 1).sum()
num_datapoints_label_2 = (dataset_dataframe['label'] == 2).sum()
num_datapoints_label_3 = (dataset_dataframe['label'] == 3).sum()
plotfigure = go.Figure(go.Bar(x=['Label = 1', 'Label =2','Label =3'], y=[num_datapo
plotfigure.update_layout(title='Number of Instances with Label = 1 vs. Label = 2 vs
```



In [14]:
```
print("Number of datapoints with Label = 1 is : ")
```

```
dataset_dataframe[dataset_dataframe['label'] == 1].shape[0]
```

```
Number of datapoints with Label = 1 is :
```

Out[14]: 679

In [15]:
```
print("Number of datapoints with Label = 2 is : ")
dataset_dataframe[dataset_dataframe['label'] == 2].shape[0]
```

```
Number of datapoints with Label = 2 is :
```

Out[15]: 470

In [16]:
```
print("Number of datapoints with Label = 3 is : ")
dataset_dataframe[dataset_dataframe['label'] == 3].shape[0]
```

```
Number of datapoints with Label = 3 is :
```

Out[16]: 845

Observation

The dataset is somewhat balanced but imbalance exists to a certain extent in the dataset after discretization.

In [17]:
```
dataset_dataframe[dataset_dataframe['label'] == 1].describe()
```

Out[17]:

	population	householdsize	racepctblack	racePctWhite	racePctAsian	racePctHisp	a
count	679.000000	679.000000	679.000000	679.000000	679.000000	679.000000	
mean	0.019764	0.483520	0.044728	0.917349	0.132680	0.045243	
std	0.024385	0.148192	0.073164	0.085830	0.164321	0.076717	
min	0.000000	0.070000	0.000000	0.390000	0.010000	0.000000	
25%	0.000000	0.380000	0.010000	0.890000	0.040000	0.010000	
50%	0.010000	0.460000	0.020000	0.950000	0.070000	0.020000	
75%	0.030000	0.560000	0.050000	0.970000	0.150000	0.050000	
max	0.160000	1.000000	0.720000	1.000000	1.000000	1.000000	

8 rows × 102 columns

Observation

The statistical measures of the datapoints belonging to class with label 1.

In [18]:
```
dataset_dataframe[dataset_dataframe['label'] == 2].describe()
```

Out[18]:

	population	householdsize	racepctblack	racePctWhite	racePctAsian	racePctHisp	a
count	470.000000	470.000000	470.000000	470.000000	470.000000	470.000000	

mean	0.036319	0.444830	0.117191	0.816681	0.159766	0.118191
std	0.054502	0.148854	0.167030	0.166057	0.211910	0.173003
min	0.000000	0.030000	0.000000	0.000000	0.000000	0.000000
25%	0.010000	0.340000	0.020000	0.752500	0.040000	0.020000
50%	0.020000	0.425000	0.050000	0.870000	0.080000	0.050000
75%	0.040000	0.530000	0.140000	0.930000	0.180000	0.140000
max	0.620000	1.000000	1.000000	1.000000	1.000000	1.000000

8 rows × 102 columns

Observation

The statistical measures of the datapoints belonging to class with label 2.

In [19]:
```
dataset_dataframe[dataset_dataframe['label'] == 3].describe()
```

Out[19]:

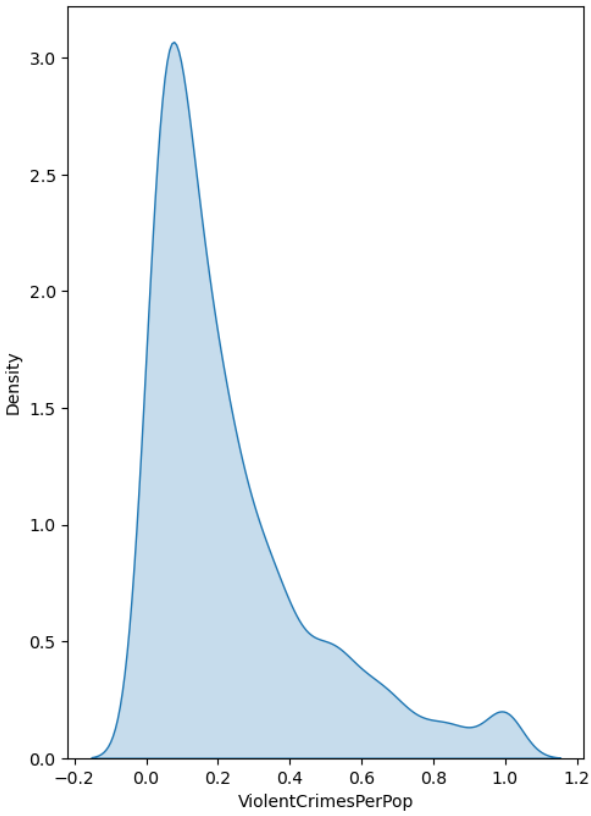
	population	householdsize	racepctblack	racePctWhite	racePctAsian	racePctHisp
count	845.000000	845.000000	845.000000	845.000000	845.000000	845.000000
mean	0.099822	0.457550	0.322757	0.587207	0.167172	0.237763
std	0.180868	0.181069	0.307495	0.260546	0.236272	0.298763
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.010000	0.340000	0.060000	0.420000	0.030000	0.020000
50%	0.040000	0.420000	0.210000	0.630000	0.070000	0.090000
75%	0.100000	0.530000	0.520000	0.790000	0.180000	0.350000
max	1.000000	1.000000	1.000000	0.990000	1.000000	1.000000

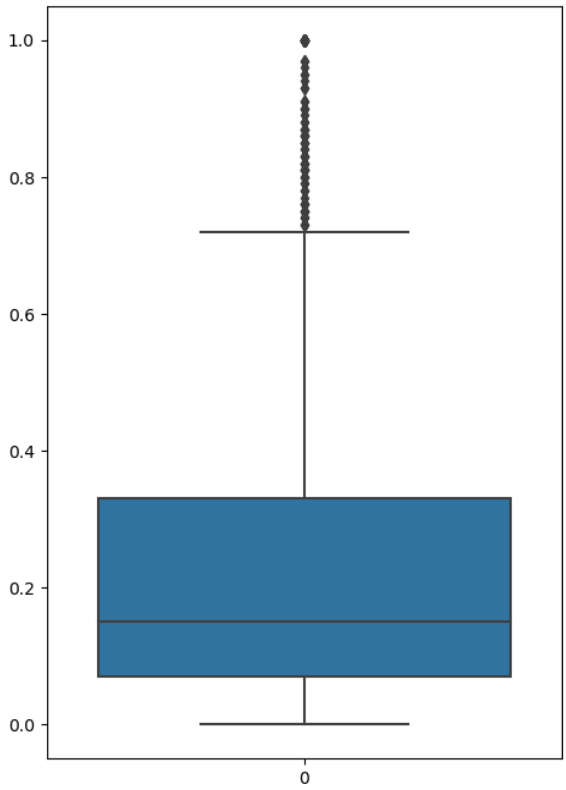
8 rows × 102 columns

Observation

The statistical measures of the datapoints belonging to class with label 3.

In [20]:
```
figure, figureaxis =plt.subplots(1,2,figsize=(12,8))
sns.kdeplot(dataset_dataframe['ViolentCrimesPerPop'],fill=True,ax=figureaxis[0])
sns.boxplot(dataset_dataframe['ViolentCrimesPerPop'],ax=figureaxis[1])
figure.show()
```





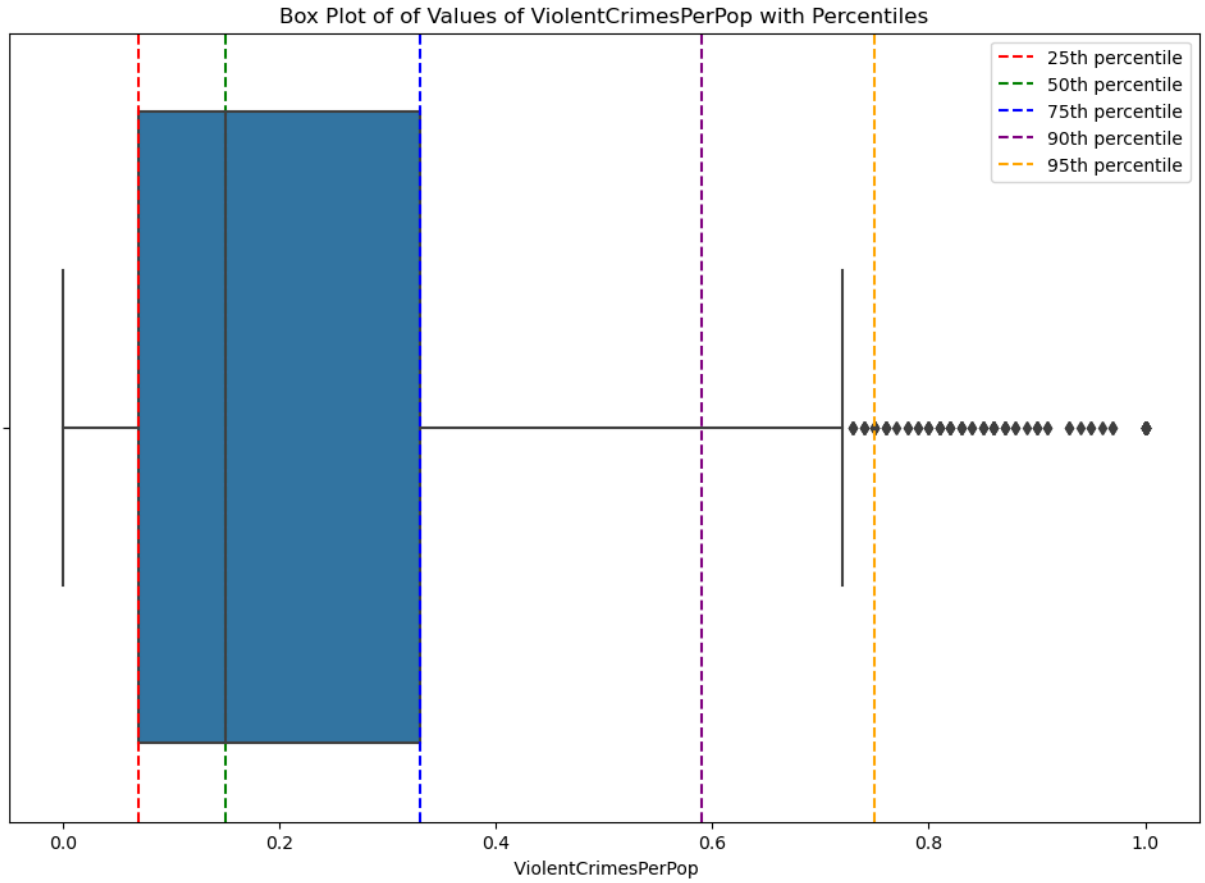
Observation

The distribution of the target ViolentCrimesPerPop is slightly skewed and Gaussian in nature.

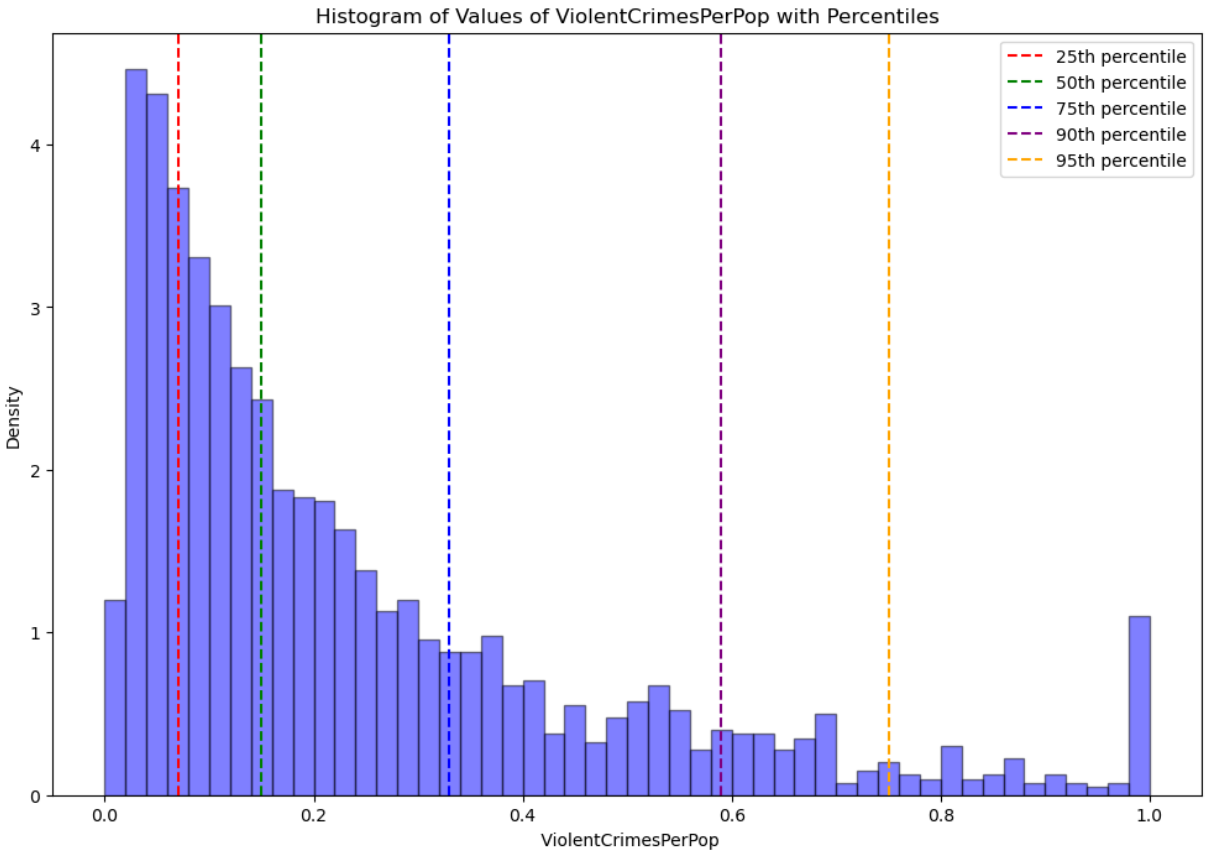
In [21]:
```
viocrperpop = dataset_dataframe['ViolentCrimesPerPop']

percentiles = [25, 50, 75, 90, 95]
percentile_values = np.percentile(viocrperpop, percentiles)

# Display percentile values
for i, p in enumerate(percentiles):
    print(f"{p}th percentile: {percentile_values[i]}")
```

```
25th percentile: 0.07
50th percentile: 0.15
75th percentile: 0.33
90th percentile: 0.59
95th percentile: 0.75
```

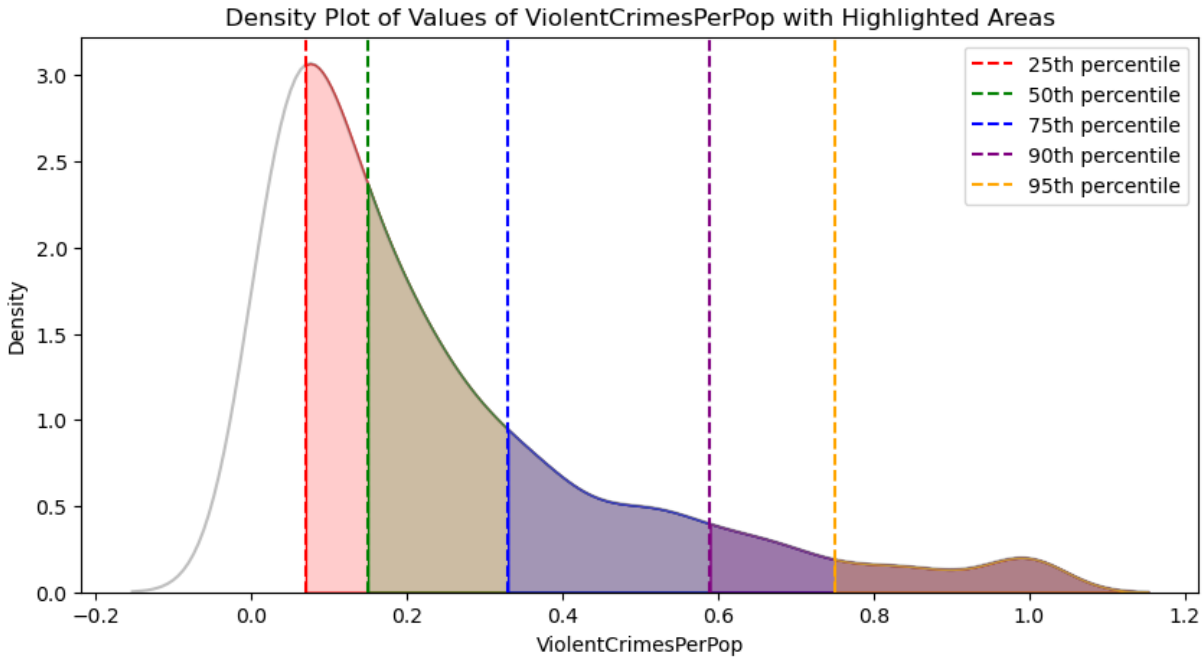
In [22]:
```
colors = ['red', 'green', 'blue', 'purple', 'orange']

# Create a box plot
plt.figure(figsize=(12, 8))
sns.boxplot(x=viocrperpop)
for p, value, color in zip(percentiles, percentile_values, colors):
    plt.axvline(value, color=color, linestyle='--', label=f'{p}th percentile')
plt.xlabel(' ViolentCrimesPerPop')
plt.legend()
plt.title('Box Plot of of Values of ViolentCrimesPerPop with Percentiles')
plt.show()
```



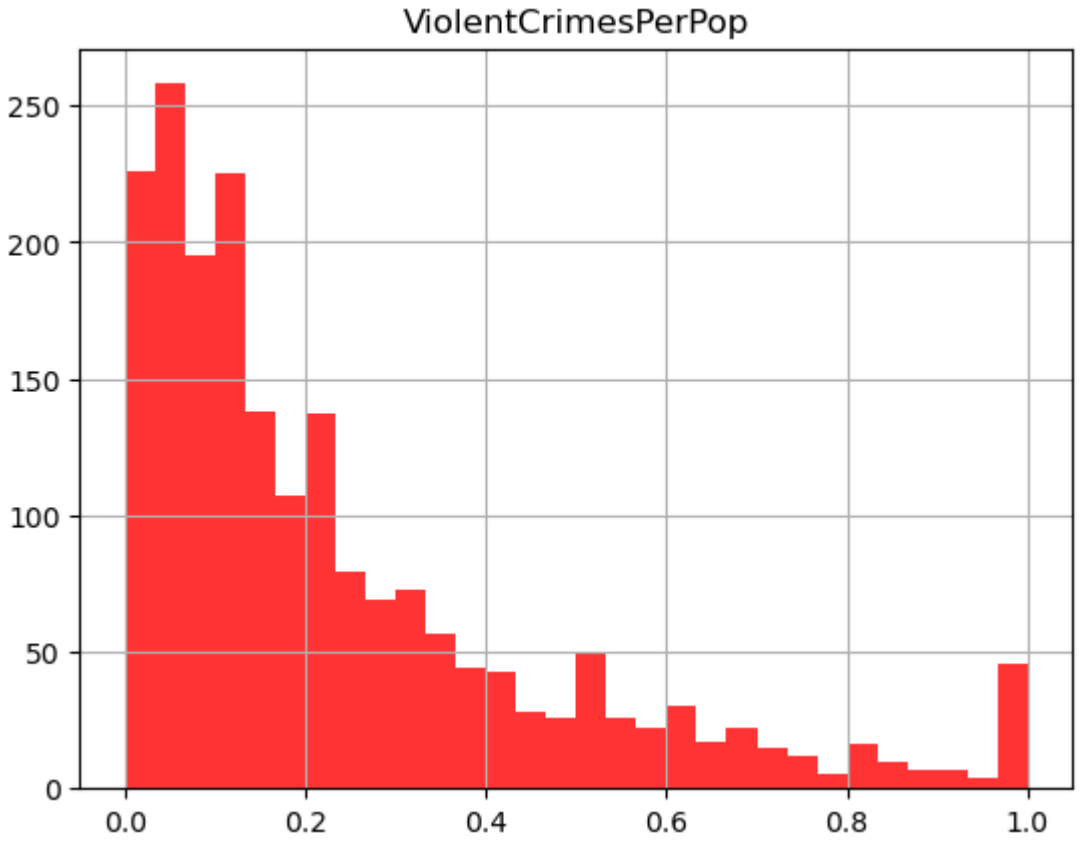
In [23]:
```
plt.figure(figsize=(12, 8))
plt.hist(viocrperpop, bins=50, color='blue', edgecolor='black', alpha=0.5, density=
for p, value, color in zip(percentiles, percentile_values, colors):
    plt.axvline(value, color=color, linestyle='--', label=f'{p}th percentile')
plt.xlabel(' ViolentCrimesPerPop')
plt.ylabel('Density')
plt.legend()
plt.title('Histogram of Values of ViolentCrimesPerPop with Percentiles')
plt.show()
```



In [24]:
```
plt.figure(figsize=(10, 5))
sns.kdeplot(viocrperpop, fill=False, color='gray', alpha=0.5)

for p, value, color in zip(percentiles, percentile_values, colors):
    plt.axvline(value, color=color, linestyle='--', label=f'{p}th percentile')
    sns.kdeplot(viocrperpop, fill=True, clip=(value, np.inf), color=color, alpha=0.

plt.xlabel('ViolentCrimesPerPop')
plt.ylabel('Density')
plt.legend()
plt.title('Density Plot of Values of ViolentCrimesPerPop with Highlighted Areas')
plt.show()
```



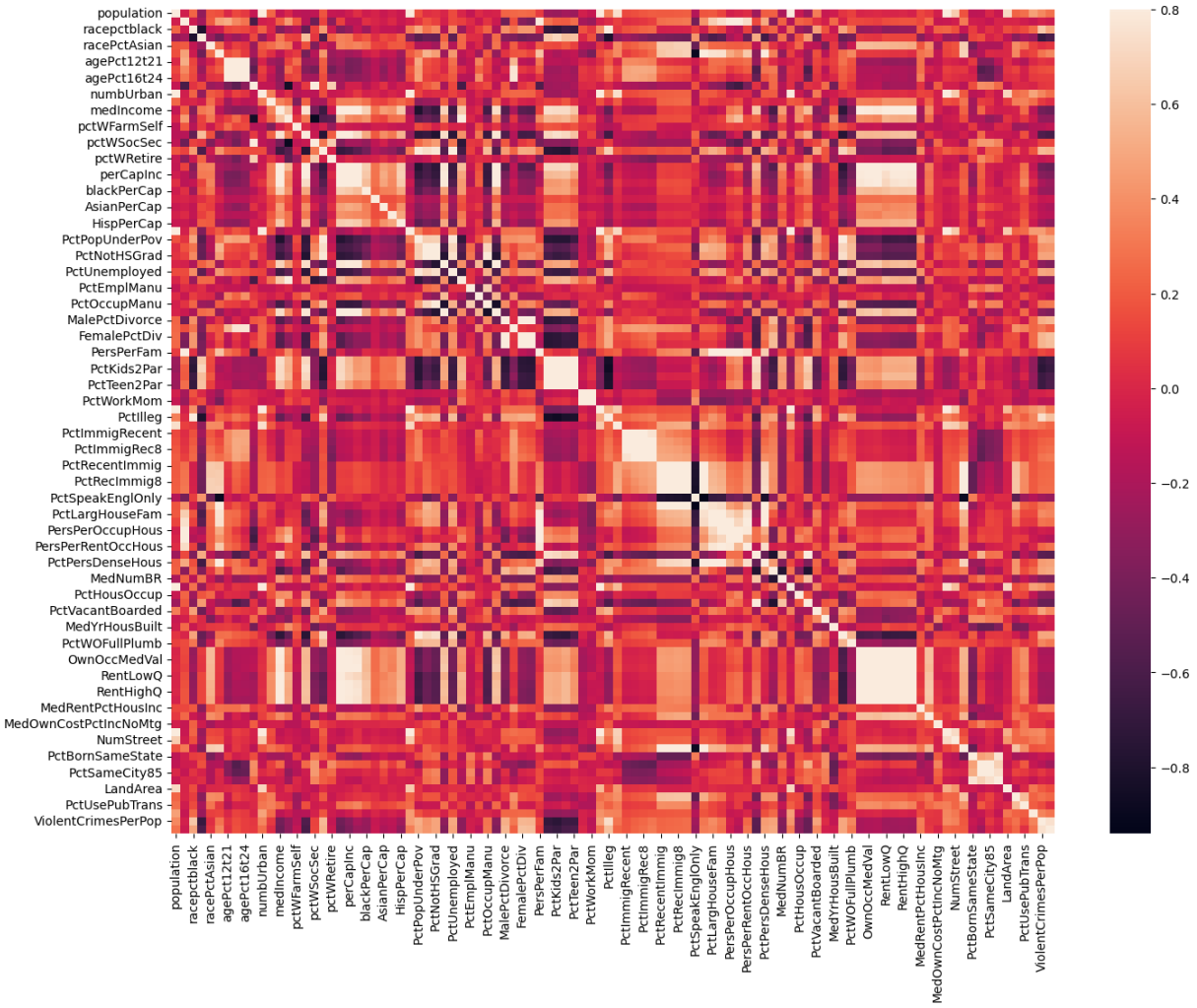
In [25]:
```
dataset_dataframe.hist(column = ['ViolentCrimesPerPop'], bins = 30, color = 'red',
plt.show()
```



Observation

From the above graphs we can observe the median, 25th,75th percentiles. There are a large number of small values and less number of bigger values.

In [26]:
```
corrmat = dataset_dataframe.corr()
fig = plt.figure(figsize = (16, 12))

sns.heatmap(corrmat, vmax = 0.8)
plt.show()
```



In [27]:
```
corrT = dataset_dataframe.corr(method = 'pearson').round(4)
corrT = corrT.sort_values(by=['ViolentCrimesPerPop'])
corrT['ViolentCrimesPerPop']
```

Out[27]:
```
PctKids2Par           -0.7384
PctFam2Par            -0.7067
racePctWhite          -0.6848
PctYoungKids2Par      -0.6661
PctTeen2Par           -0.6616
                         ...
pctWPubAsst            0.5747
racepctblack           0.6313
PctIlleg               0.7380
label                  0.7486
ViolentCrimesPerPop    1.0000
Name: ViolentCrimesPerPop, Length: 102, dtype: float64
```

Observation

The correlations between any pairs of features are not very high.

Dimensionality Reduction

Since there are 102 features, we will perform dimensionality reduction and use only 16 features for our ML models.

Principal Component Analysis

In this section we have implemented the Principal Component Analysis from scratch. Principal Component Analysis is a dimensionality reduction technique which uses the concept of eigen values and variance to reduce the number of dimensions or features of the dataset.

In [28]:
```
class PCA1:
    def __init__(self, n_components):
        self.n_components = n_components
        self.mean = None
        self.components = None
        self.explained_variance = None
        self.explained_variance_ratio = None

    def fit(self, X):
        # Calculate the mean of the data
        self.mean = np.mean(X, axis=0)

        # Center the data by subtracting the mean
        X_centered = X - self.mean

        # Compute the covariance matrix
        cov_matrix = np.cov(X_centered, rowvar=False)

        # Eigenvalue decomposition of the covariance matrix
        eigenvalues, eigenvectors = np.linalg.eigh(cov_matrix)

        # Sort eigenvalues and eigenvectors in descending order
        sorted_indices = np.argsort(eigenvalues)[::-1]
        eigenvalues = eigenvalues[sorted_indices]
        eigenvectors = eigenvectors[:, sorted_indices]

        # Select the top n_components eigenvectors
        self.components = eigenvectors[:, :self.n_components]

        # Calculate explained variance and explained variance ratio
        total_variance = np.sum(eigenvalues)
        self.explained_variance = eigenvalues[:self.n_components]
        self.explained_variance_ratio = self.explained_variance / total_variance

    def transform(self, X):
        # Center the data by subtracting the mean
        X_centered = X - self.mean
```

```
        # Project the data onto the selected principal components
        return np.dot(X_centered, self.components)
```

In [29]:
```
pca = PCA1(n_components = 16)
pca.fit(X)
X_reduced = pca.transform(X)
```

In [30]:
```
X_reduced.shape[1]
```

Out[30]: 16

Observation

After performing dimensionality reduction using PCA we can see that the number of features have reduced from 101 to 16.

Data Preparation

In [31]:
```
std = StandardScaler()
std.fit(X_reduced)
X_std = std.fit_transform(X_reduced)
```

Observation

Since the scales of different features are different we are performing standardization and bringing the values between 0 and 1.

In [32]:
```
X_train, X_temp, y_train, y_temp = train_test_split(X_std, y_data, test_size=0.40,

# Split the temp dataset into cross-validation and test
X_validation, X_test, y_validation, y_test = train_test_split(X_temp, y_temp, test_
```

Observation

We are dividing the entire dataset into the following three groups:

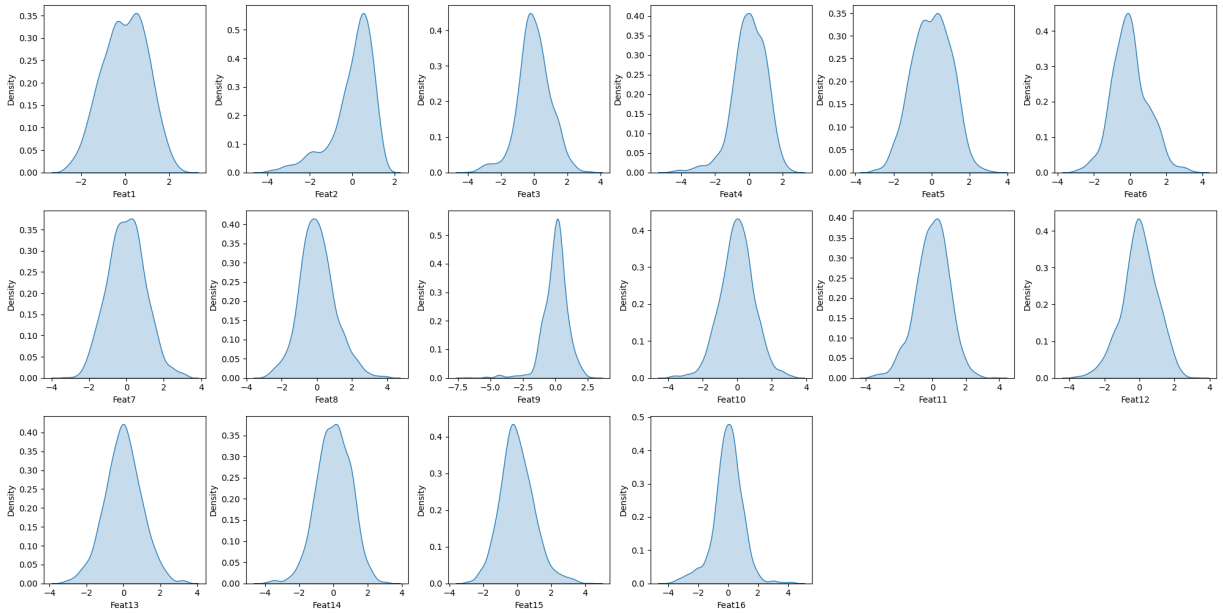
1) 60 % of the dataset are training data used for training the ML models.
2) 20 % of the dataset are validation data used for choosing the most appropriate hyperparameter.
3) 20 % of the dataset are test data used for evaluating the performance of the model using the unseen dataset.

In [33]:
```
columns = ['Feat1', 'Feat2', 'Feat3','Feat4','Feat5','Feat6','Feat7','Feat8','Feat9

# Convert 2D array to DataFrame
train_dataframe = pd.DataFrame(X_train, columns=columns)
train_dataframe['label'] = y_train
test_dataframe = pd.DataFrame(X_test, columns=columns)
```

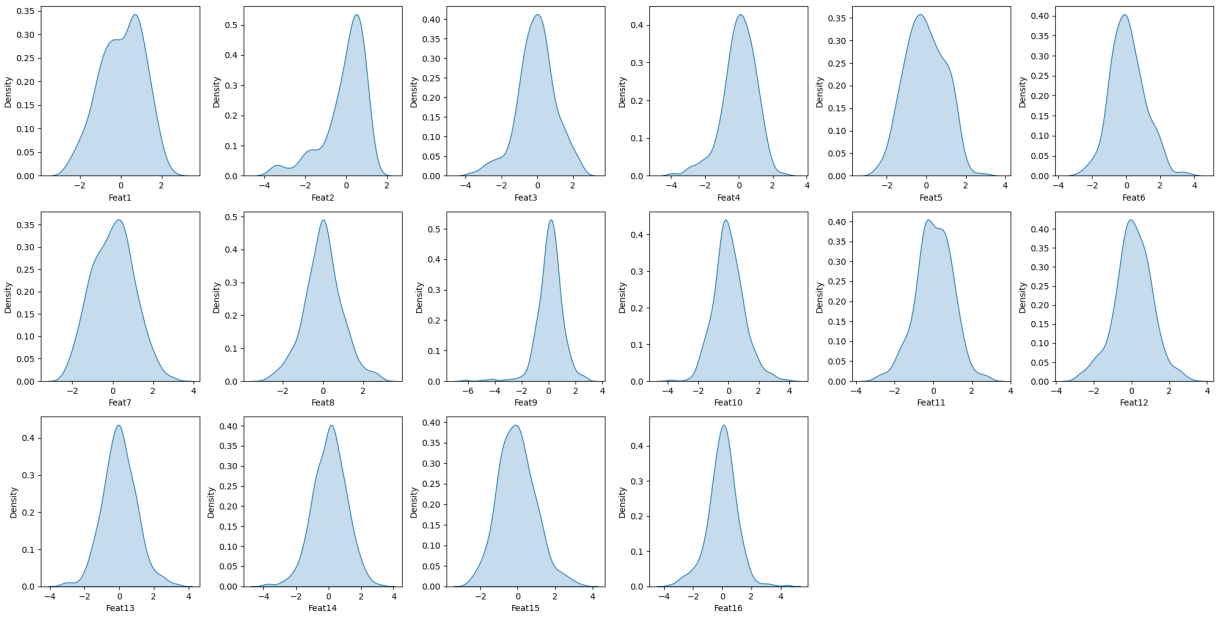
```
test_dataframe['label'] = y_test
validation_dataframe = pd.DataFrame(X_validation, columns=columns)
validation_dataframe['label'] = y_validation
```

Exploratory Data Analysis of Data to be given to Models

In [34]:
```
plt.figure(figsize=(20,40))
for i,feat in enumerate(columns,1):
    plt.subplot(12,6,i)
    sns.kdeplot(train_dataframe[feat],fill=True)
plt.tight_layout()
plt.show()
```



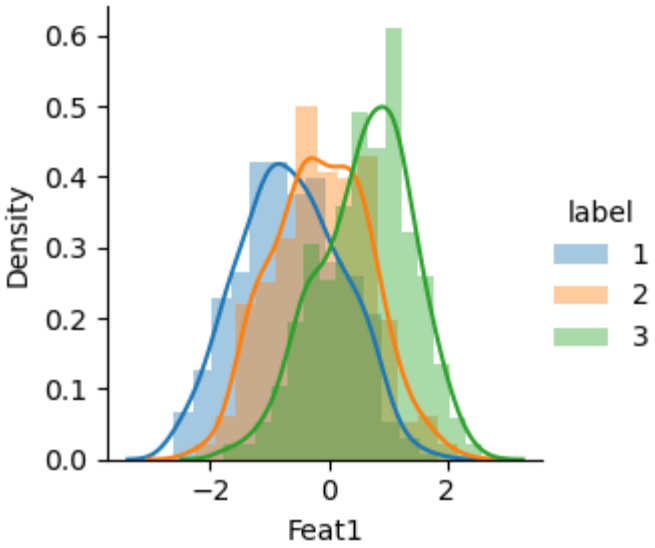
In [35]:
```
plt.figure(figsize=(20,40))
for i,feat in enumerate(columns,1):
    plt.subplot(12,6,i)
    sns.kdeplot(test_dataframe[feat],fill=True)
plt.tight_layout()
plt.show()
```



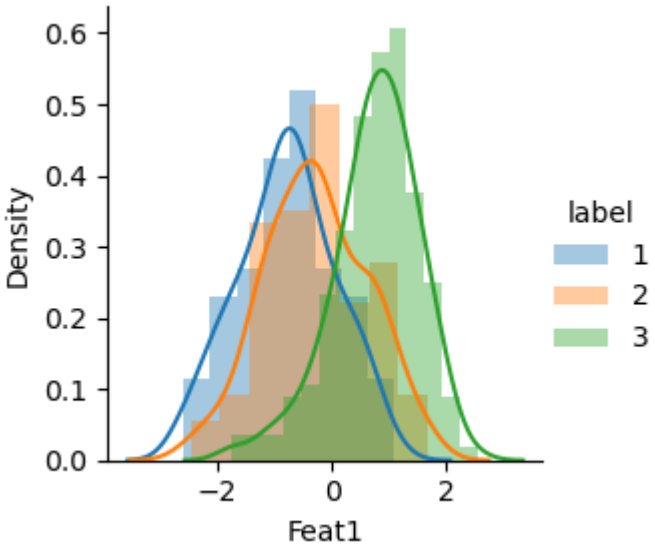
Observation

Most of the features follow almost Gaussian distribution and the skewness of the data are very very less.

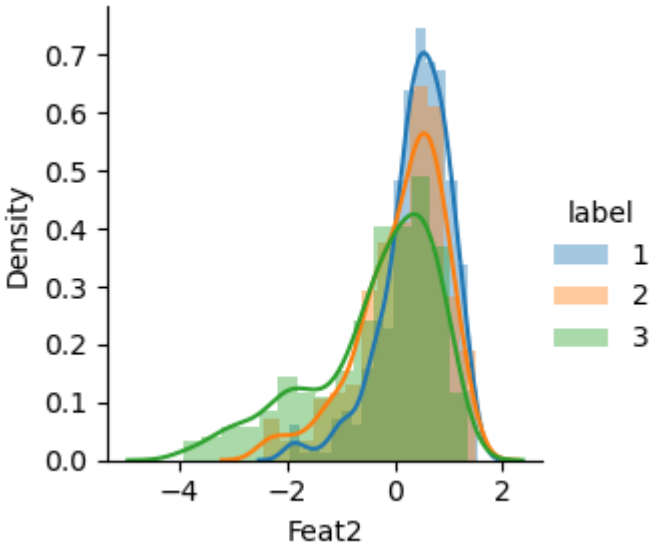
In [36]:
```
sns.FacetGrid(train_dataframe,hue="label")\
   .map(sns.distplot,"Feat1")\
   .add_legend();
plt.show()
```



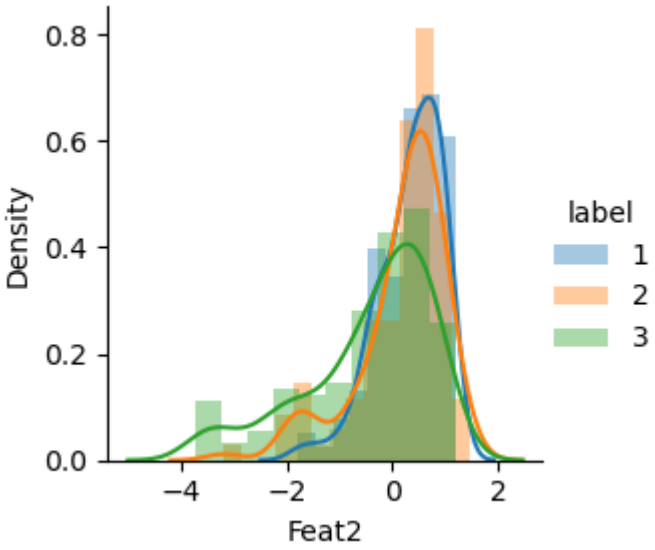
In [37]:
```
sns.FacetGrid(test_dataframe,hue="label")\
   .map(sns.distplot,"Feat1")\
   .add_legend();
plt.show()
```



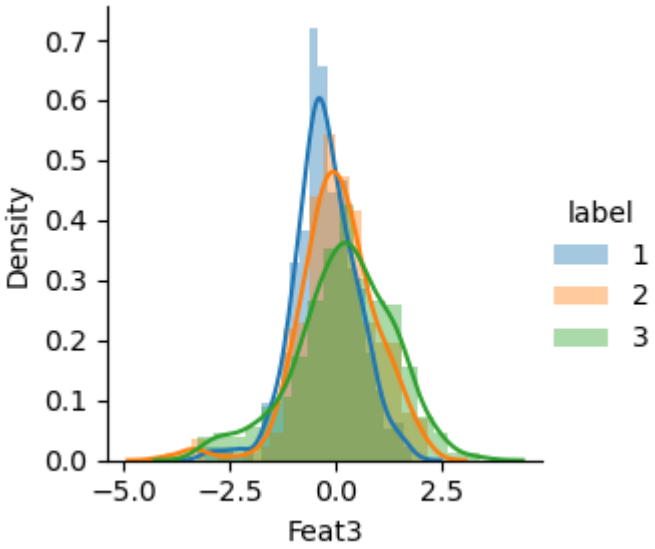
In [38]:
```
sns.FacetGrid(train_dataframe,hue="label")\
   .map(sns.distplot,"Feat2")\
   .add_legend();
plt.show()
```



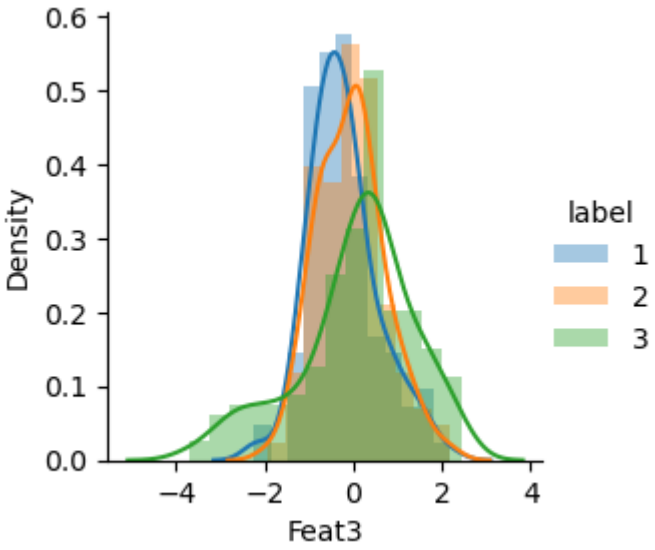
In [39]:
```
sns.FacetGrid(test_dataframe,hue="label")\
   .map(sns.distplot,"Feat2")\
   .add_legend();
plt.show()
```



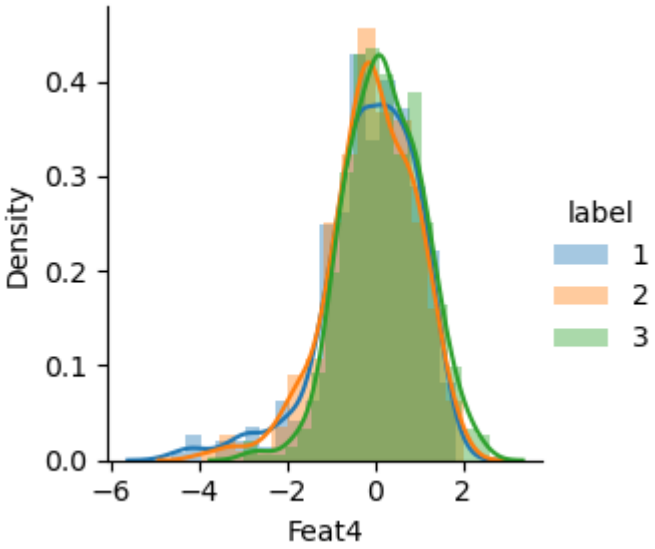
In [40]:
```
sns.FacetGrid(train_dataframe,hue="label")\
   .map(sns.distplot,"Feat3")\
   .add_legend();
plt.show()
```



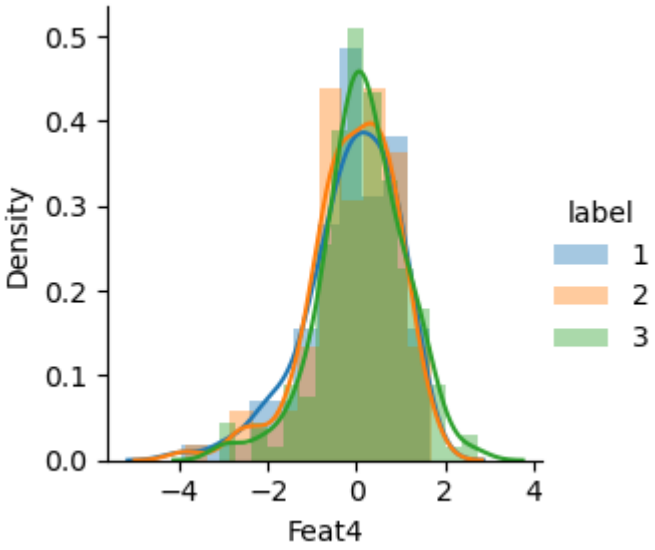
In [41]:
```
sns.FacetGrid(test_dataframe,hue="label")\
   .map(sns.distplot,"Feat3")\
   .add_legend();
plt.show()
```



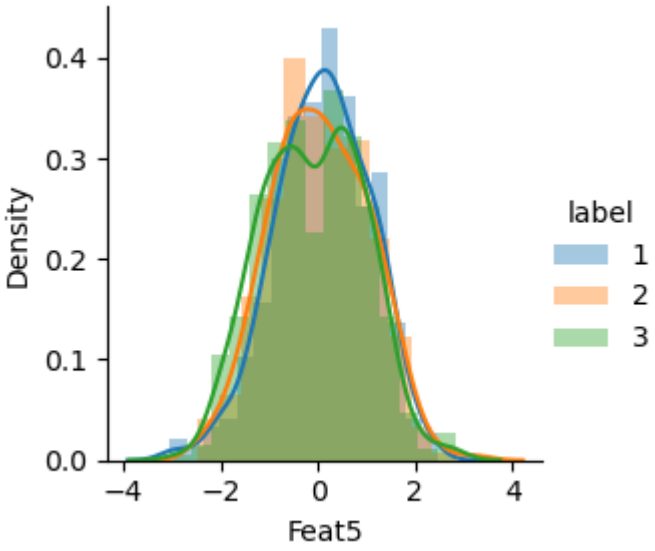
In [42]:
```
sns.FacetGrid(train_dataframe,hue="label")\
   .map(sns.distplot,"Feat4")\
   .add_legend();
plt.show()
```



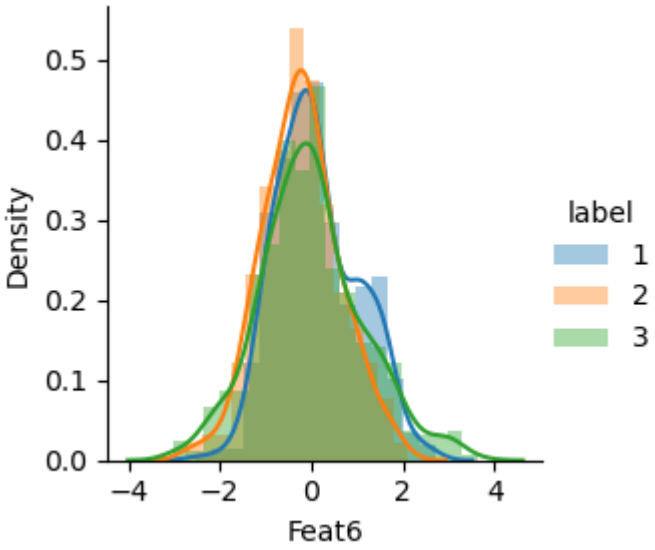
In [43]:
```
sns.FacetGrid(test_dataframe,hue="label")\
   .map(sns.distplot,"Feat4")\
   .add_legend();
plt.show()
```



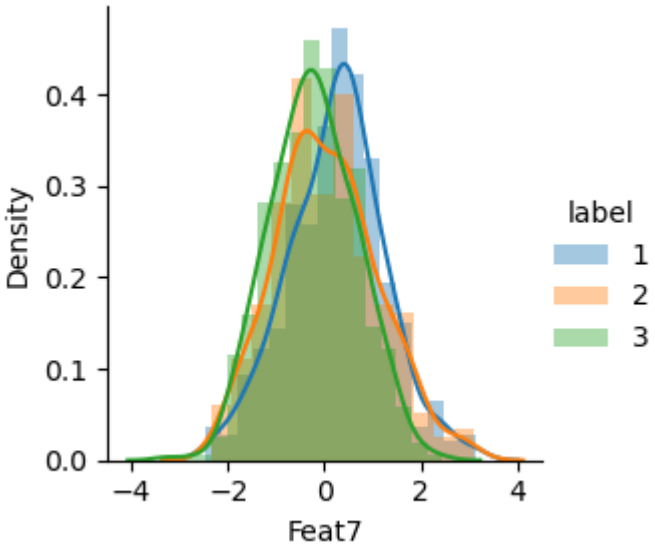
In [44]:
```
sns.FacetGrid(train_dataframe,hue="label")\
   .map(sns.distplot,"Feat5")\
   .add_legend();
plt.show()
```



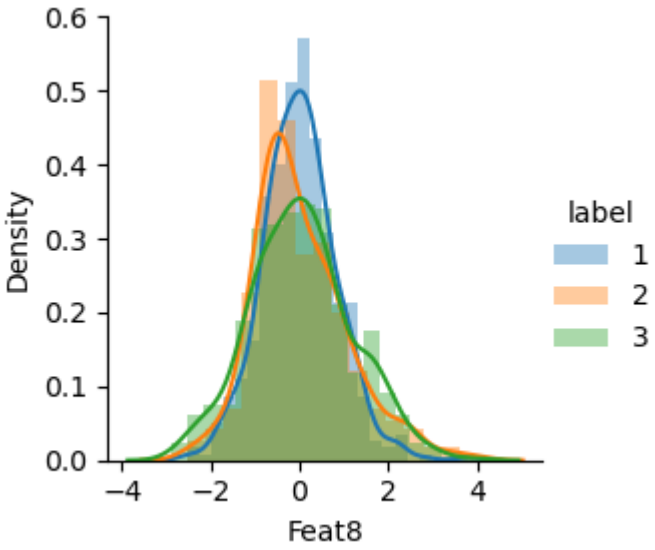
In [45]:
```
sns.FacetGrid(train_dataframe,hue="label")\
   .map(sns.distplot,"Feat6")\
   .add_legend();
plt.show()
```



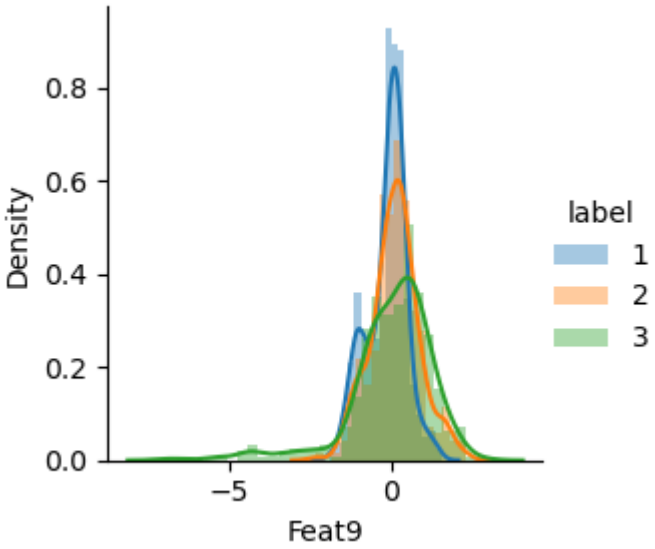
In [46]:
```
sns.FacetGrid(train_dataframe,hue="label")\
   .map(sns.distplot,"Feat7")\
   .add_legend();
plt.show()
```



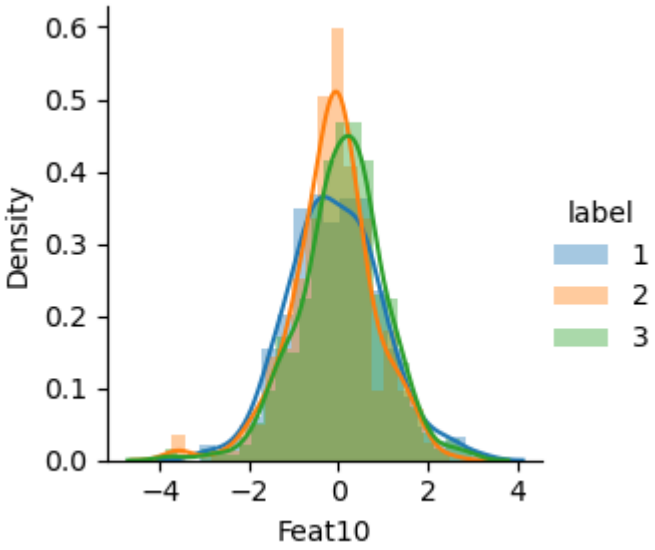
In [47]:
```
sns.FacetGrid(train_dataframe,hue="label")\
   .map(sns.distplot,"Feat8")\
   .add_legend();
plt.show()
```



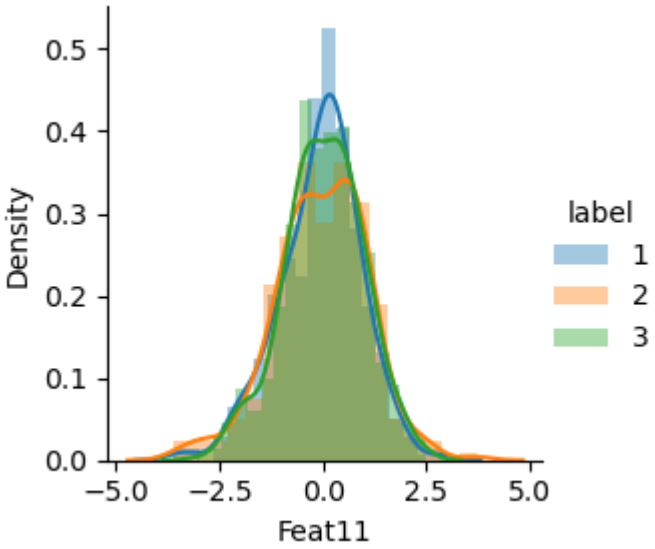
In [48]:
```
sns.FacetGrid(train_dataframe,hue="label")\
   .map(sns.distplot,"Feat9")\
   .add_legend();
plt.show()
```



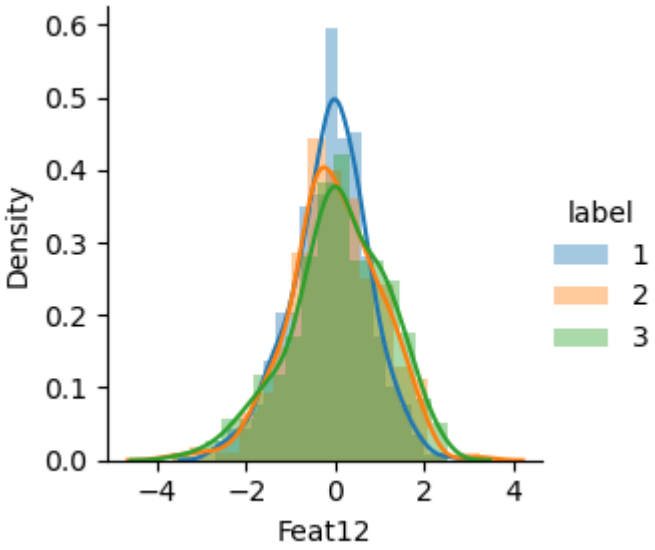
In [49]:
```
sns.FacetGrid(train_dataframe,hue="label")\
   .map(sns.distplot,"Feat10")\
   .add_legend();
plt.show()
```



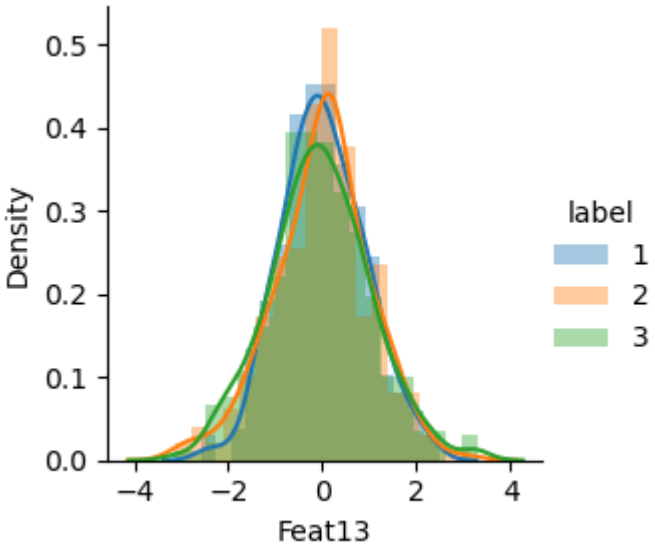
In [50]:
```
sns.FacetGrid(train_dataframe,hue="label")\
   .map(sns.distplot,"Feat11")\
   .add_legend();
plt.show()
```



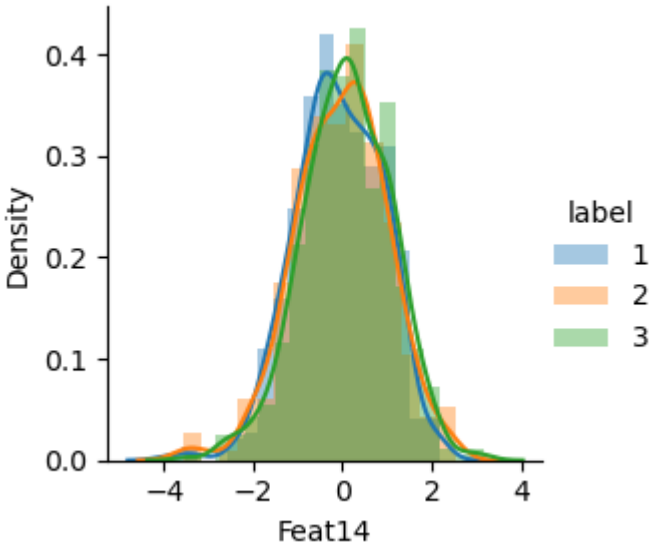
In [51]:
```
sns.FacetGrid(train_dataframe,hue="label")\
   .map(sns.distplot,"Feat12")\
   .add_legend();
plt.show()
```



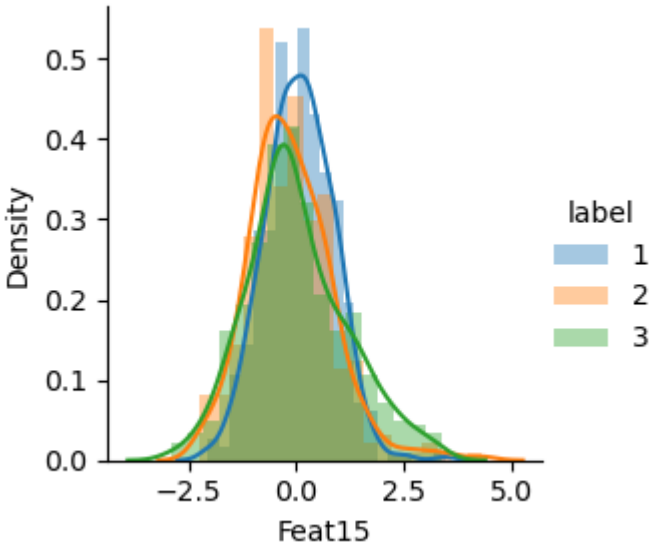
In [52]:
```
sns.FacetGrid(train_dataframe,hue="label")\
   .map(sns.distplot,"Feat13")\
   .add_legend();
plt.show()
```



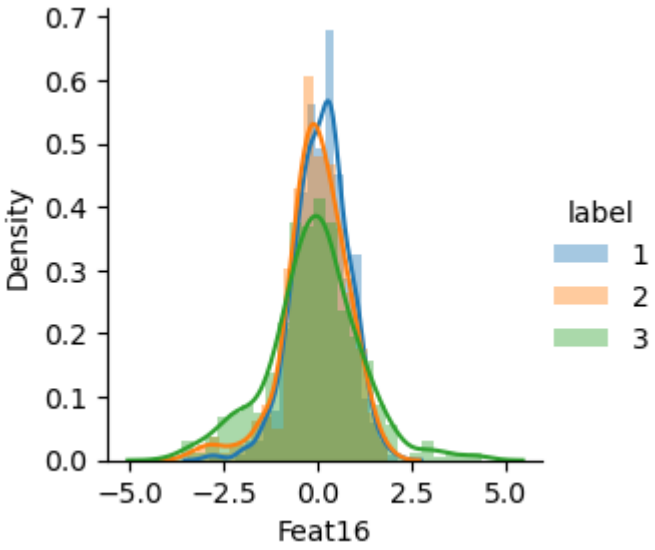
In [53]:
```
sns.FacetGrid(train_dataframe,hue="label")\
   .map(sns.distplot,"Feat14")\
   .add_legend();
plt.show()
```



In [54]:
```
sns.FacetGrid(train_dataframe,hue="label")\
   .map(sns.distplot,"Feat15")\
   .add_legend();
plt.show()
```



In [55]:
```
sns.FacetGrid(train_dataframe,hue="label")\
   .map(sns.distplot,"Feat16")\
   .add_legend();
plt.show()
```



Observation

Since the kde and histogram plots of different classes overlap with each other it indicates that the each of the independent features are not suitable for the classification of the dataset.

2. Decision tree model with entropy implementation

2.1 Implementation of the Model

In [135…
```
def perform_Data_classification(data):
    uniqueClasses, uniqueClassesCounts = numpy.unique(data[:, -1], return_counts =
    return uniqueClasses[uniqueClassesCounts.argmax()]
```

In [136…
```
def Purity_chk_bool(dataset):
    if len(numpy.unique(dataset[:, -1])) == 1:
        return True
    else:
        return False
```

In [137…
```
def splitData(data, splitColumn, splitValue):
    splitColumnValues = data[:, splitColumn]
    return data[splitColumnValues <= splitValue], data[splitColumnValues > splitVal
```

In [138…
```
def Splits_at_potential_feat(dataset, selected_attributes):
    splits_candidates = {}
    num_rows, num_columns = dataset.shape
    attribute_indices = list(range(num_columns - 1))

    if selected_attributes is not None and len(selected_attributes) <= len(attribut
```

```
        attribute_indices = selected_attributes

    for attribute_index in attribute_indices:
        attribute_values = dataset[:, attribute_index]
        unique_attribute_values = np.unique(attribute_values)

        if len(unique_attribute_values) == 1:
            splits_candidates[attribute_index] = unique_attribute_values
        else:
            splits_candidates[attribute_index] = []
            for i in range(len(unique_attribute_values)):
                if i != 0:
                    current_val = unique_attribute_values[i]
                    prev_val = unique_attribute_values[i - 1]
                    splits_candidates[attribute_index].append((current_val + prev_v

    return splits_candidates
```

In [139…

```
def calculateEntropy(data):
    _, uniqueClassesCounts = numpy.unique(data[:, -1], return_counts = True)
    probabilities = uniqueClassesCounts / uniqueClassesCounts.sum()
    return sum(probabilities * -numpy.log2(probabilities))
```

In [140…

```
def calculateOverallEntropy(dataBelow, dataAbove):
    pDataBelow = len(dataBelow) / (len(dataBelow) + len(dataAbove))
    pDataAbove = len(dataAbove) / (len(dataBelow) + len(dataAbove))
    return pDataBelow * calculateEntropy(dataBelow) + pDataAbove * calculateEntropy
```

In [141…

```
def determineBestSplit(data_set, potential_partitions, random_partitions=None):
    best_entropy_so_far = 9999
    optimum_partition_column = 0
    optimum_partition_value = 0

    if random_partitions is None:
        for partition_column in potential_partitions:
            for partition_value in potential_partitions[partition_column]:
                below_data, above_data = splitData(data_set, partition_column, part
                current_entropy = calculateOverallEntropy(below_data, above_data)
                if current_entropy <= best_entropy_so_far:
                    best_entropy_so_far = current_entropy
                    optimum_partition_column = partition_column
                    optimum_partition_value = partition_value
    else:
        for _ in range(random_partitions):
            random_partition_column = random.choice(list(potential_partitions))
            random_partition_value = random.choice(potential_partitions[random_part
            below_data, above_data = divide_data(data_set, random_partition_column,
            current_entropy = assess_total_entropy(below_data, above_data)
            if current_entropy <= best_entropy_so_far:
                best_entropy_so_far = current_entropy
                optimum_partition_column = random_partition_column
                optimum_partition_value = random_partition_value

    return optimum_partition_column, optimum_partition_value
```

In [142…
```
def buildDecisionTree(data_frame, current_depth=0, min_sample_size=2, max_depth=100
    if current_depth == 0:
        global COLUMN_HEADERS
        COLUMN_HEADERS = data_frame.columns
        data = data_frame.values
        if random_attributes is not None and random_attributes <= len(COLUMN_HEADER
            random_attributes = random.sample(population=list(range(len(COLUMN_HEAD
        else:
            random_attributes = None
    else:
        data = data_frame

    if Purity_chk_bool(data) or len(data) < min_sample_size or current_depth == max
        return perform_Data_classification(data)
    else:
        current_depth += 1
        potential_splits = Splits_at_potential_feat(data, random_attributes)
        split_column, split_value = determineBestSplit(data, potential_splits, rand
        data_below, data_above = splitData(data, split_column, split_value)

        if len(data_below) == 0 or len(data_above) == 0:
            return perform_Data_classification(data)
        else:
            question = str(COLUMN_HEADERS[split_column]) + " <= " + str(split_value
            decision_sub_tree = {question: []}
            yes_answer = buildDecisionTree(data_below, current_depth, min_sample_si
            no_answer = buildDecisionTree(data_above, current_depth, min_sample_siz

            if yes_answer == no_answer:
                decision_sub_tree = yes_answer
            else:
                decision_sub_tree[question].append(yes_answer)
                decision_sub_tree[question].append(no_answer)

            return decision_sub_tree
```

In [143…
```
def classifySample(sample, decisionTree):
    if not isinstance(decisionTree, dict):
        return decisionTree
    question = list(decisionTree.keys())[0]
    attribute, value = question.split(" <= ")
    if sample[attribute] <= float(value):
        answer = decisionTree[question][0]
    else:
        answer = decisionTree[question][1]
    return classifySample(sample, answer)
```

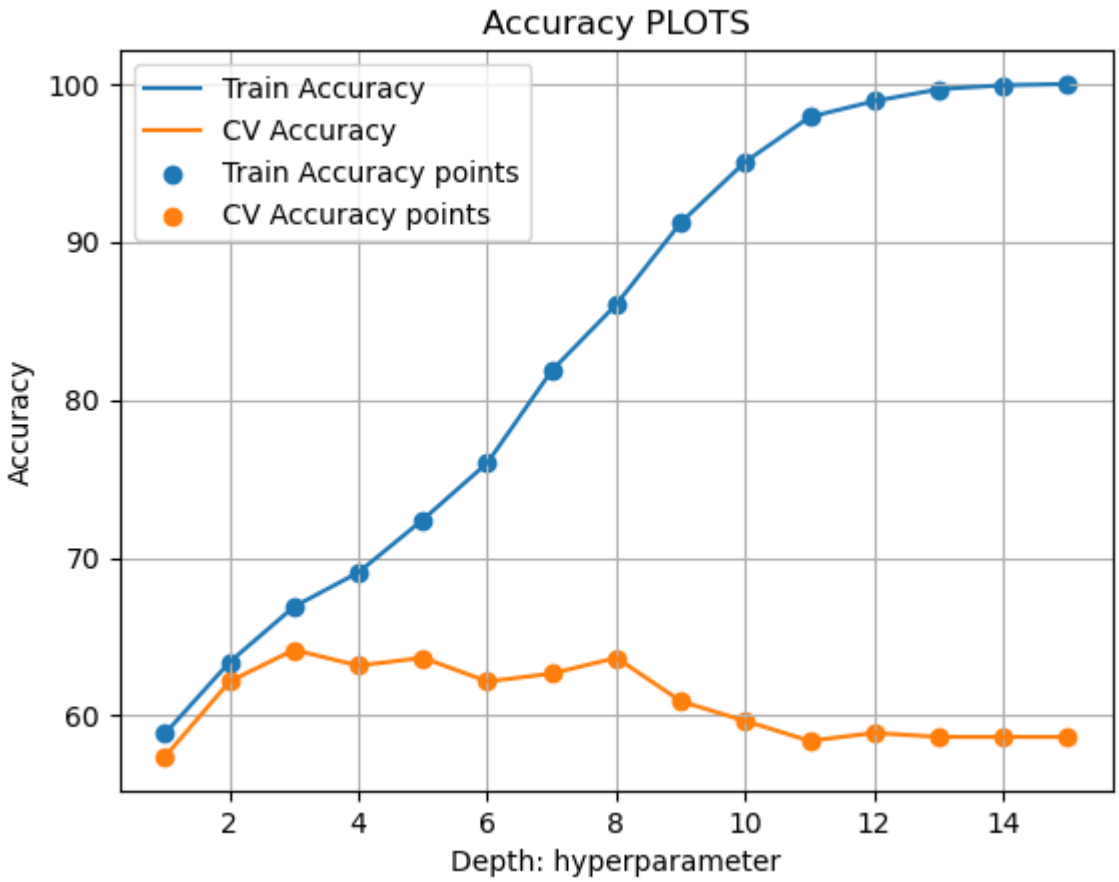
In [144…
```
def decisionTreePredictions(dataFrame, decisionTree):
    predictions = dataFrame.apply(classifySample, axis = 1, args = (decisionTree,))
    return predictions
```

In [145…
```
def calculateAccuracy(predictedResults, category):
    resultCorrect = predictedResults == category
    return resultCorrect.mean()
```

In [146…
```
i = 1
train_acc = []
cv_acc = []
depth = []
accuracyTrain = 0
while accuracyTrain < 100:
    decisionTree = buildDecisionTree(train_dataframe, max_depth = i)
    decisionTreeTestResults = decisionTreePredictions(validation_dataframe, decisio
    accuracyCV = calculateAccuracy(decisionTreeTestResults, validation_dataframe.il
    cv_acc.append(accuracyCV)
    decisionTreeTrainResults = decisionTreePredictions(train_dataframe, decisionTre
    accuracyTrain = calculateAccuracy(decisionTreeTrainResults, train_dataframe.ilo
    train_acc.append(accuracyTrain)
    print("\n maxDepth = {}: ".format(i), end = "")
    print("accuracy of CV = {0:.2f}%, ".format(accuracyCV), end = "")
    print("accuracy of Train = {0:.2f}%, ".format(accuracyTrain), end = "")
    depth.append(i)
    i += 1
```

```
maxDepth = 1: accuracy of CV = 57.39%, accuracy of Train = 58.86%,
maxDepth = 2: accuracy of CV = 62.16%, accuracy of Train = 63.38%,
maxDepth = 3: accuracy of CV = 64.16%, accuracy of Train = 66.89%,
maxDepth = 4: accuracy of CV = 63.16%, accuracy of Train = 69.06%,
maxDepth = 5: accuracy of CV = 63.66%, accuracy of Train = 72.41%,
maxDepth = 6: accuracy of CV = 62.16%, accuracy of Train = 76.00%,
maxDepth = 7: accuracy of CV = 62.66%, accuracy of Train = 81.86%,
maxDepth = 8: accuracy of CV = 63.66%, accuracy of Train = 86.04%,
maxDepth = 9: accuracy of CV = 60.90%, accuracy of Train = 91.22%,
maxDepth = 10: accuracy of CV = 59.65%, accuracy of Train = 95.07%,
maxDepth = 11: accuracy of CV = 58.40%, accuracy of Train = 97.91%,
maxDepth = 12: accuracy of CV = 58.90%, accuracy of Train = 98.91%,
maxDepth = 13: accuracy of CV = 58.65%, accuracy of Train = 99.67%,
maxDepth = 14: accuracy of CV = 58.65%, accuracy of Train = 99.92%,
maxDepth = 15: accuracy of CV = 58.65%, accuracy of Train = 100.00%,
```

In [149…
```
plt.plot(depth, train_acc, label='Train Accuracy')
plt.plot(depth, cv_acc, label='CV Accuracy')

plt.scatter(depth, train_acc, label='Train Accuracy points')
plt.scatter(depth, cv_acc, label='CV Accuracy points')

plt.legend()
plt.xlabel("Depth: hyperparameter")
plt.ylabel("Accuracy")
plt.title("Accuracy PLOTS")
plt.grid()
plt.show()
```



In [150…
```
decisionTree = buildDecisionTree(test_dataframe, max_depth = 5)
decisionTreeTestdataResults = decisionTreePredictions(test_dataframe, decisionTree)
accuracytest = calculateAccuracy(decisionTreeTestdataResults, test_dataframe.iloc[:
```

In [151…
```
print(accuracytest)
```

```
79.9498746867168
```

In [152…
```
y_test
```

Out[152]:
```
[3,
 3,
 2,
 2,
 2,
 2,
 2,
 1,
 2,
 3,
 3,
 1,
 2,
 2,
 2,
 2,
 1,
 1,
```

3,
1,
2,
3,
3,
3,
3,
1,
1,
1,
3,
2,
3,
2,
1,
1,
3,
3,
3,
1,
3,
2,
1,
1,
3,
3,
3,
2,
2,
2,
1,
2,
2,
2,
2,
3,
1,
2,
3,
1,
3,
1,
2,
3,
2,
3,
3,
1,
3,
1,
2,
1,
3,
2,
2,
2,

3,
1,
3,
2,
3,
3,
1,
3,
1,
1,
3,
2,
3,
3,
3,
3,
3,
2,
3,
2,
3,
1,
2,
1,
3,
2,
3,
1,
3,
3,
3,
2,
1,
3,
1,
3,
1,
2,
3,
3,
3,
1,
3,
3,
3,
3,
1,
1,
1,
3,
3,
1,
1,
3,
1,
1,
1,
1,

1,
3,
3,
2,
2,
3,
3,
3,
3,
3,
3,
3,
1,
1,
3,
2,
3,
3,
1,
1,
3,
3,
3,
1,
1,
1,
1,
3,
3,
2,
3,
3,
3,
2,
2,
3,
2,
2,
3,
1,
3,
3,
1,
1,
2,
2,
2,
3,
1,
1,
1,
2,
1,
2,
1,
3,

3,
2,
3,
2,
3,
3,
2,
2,
2,
1,
2,
2,
2,
1,
2,
1,
3,
3,
3,
2,
3,
2,
3,
2,
2,
1,
3,
3,
2,
1,
2,
3,
3,
1,
2,
3,
3,
3,
1,
3,
3,
3,
1,
3,
1,
3,
3,
3,
3,
2,
2,
1,
1,
2,
2,
3,

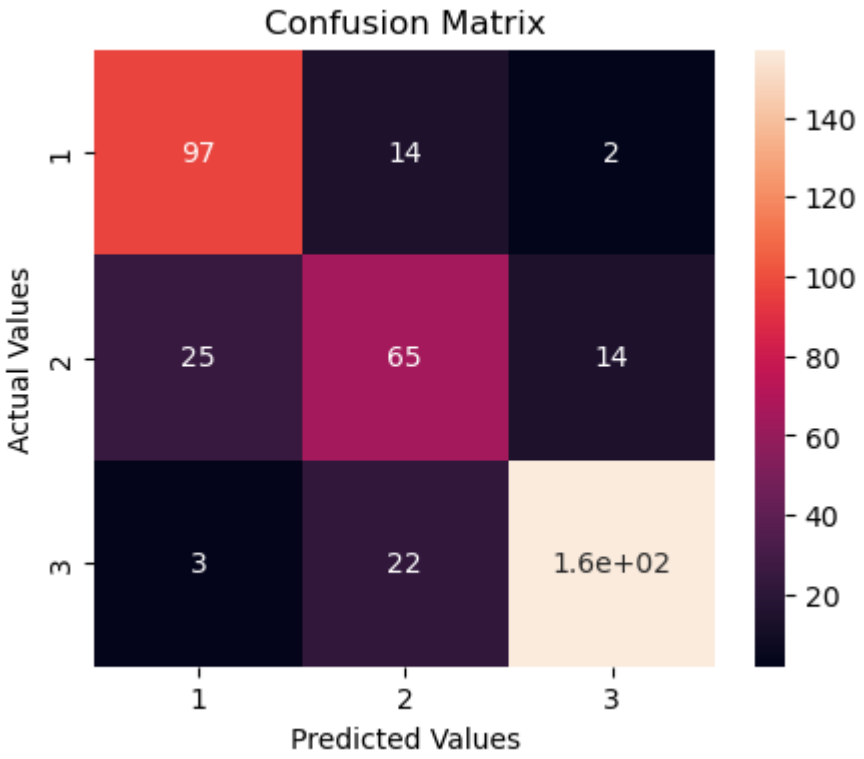
3,
2,
1,
3,
1,
2,
1,
2,
3,
2,
2,
2,
1,
1,
3,
3,
1,
3,
1,
3,
1,
1,
2,
2,
1,
2,
3,
1,
1,
2,
1,
3,
3,
3,
1,
3,
1,
3,
3,
1,
3,
3,
2,
3,
3,
1,
2,
2,
3,
3,
1,
3,
3,
2,
3,
3,

2,
2,
1,
3,
3,
3,
2,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
2,
1,
3,
2,
2,
3,
3,
3,
2,
3,
1,
3,
1,
3,
3,
3,
3,
3,
3,
3,
3,
2,
1,
1,
3,
1,
2,
1,
1,
2,
3,
2,
3,
1,
1,
1,
1,
3,

```
3,
3,
2,
3,
1,
1,
3,
2,
3,
3,
3,
3,
1,
3,
3,
2,
3,
2,
3,
2,
1,
2,
1,
3,
2,
1,
2,
1,
1,
1,
3,
1,
1,
1,
1,
1,
3,
3,
1,
3,
3,
3,
3,
2,
2]
```

In [153…

```
cm = confusion_matrix(y_test, decisionTreeTestdataResults)
cm_df = pd.DataFrame(cm,index = ['1','2','3'], columns = ['1','2','3'])
plt.figure(figsize=(5,4))
sns.heatmap(cm_df, annot=True)
plt.title('Confusion Matrix')
plt.ylabel('Actual Values')
plt.xlabel('Predicted Values')
plt.show()
```



In [154…

```
tpforclass1 = cm[0][0]
fnforclass1 = cm[0][1] + cm[0][2]
tnforclass1 = cm[1][1] + cm[1][2] + cm[2][1] + cm[2][2]
fpforclass1 = cm[1][0] + cm[2][0]

recallforclass1 = (tpforclass1) / (tpforclass1 + fnforclass1)
precisionforclass1 = (tpforclass1) / (tpforclass1 + fpforclass1)
f1scoreforclass1 = (2 * precisionforclass1 * recallforclass1) / (precisionforclass1

tpforclass2 = cm[1][1]
fnforclass2 = cm[1][0] + cm[1][2]
tnforclass2 = cm[0][0] + cm[0][2] + cm[2][0] + cm[2][2]
fpforclass2 = cm[0][1] + cm[2][1]

recallforclass2 = (tpforclass2) / (tpforclass2 + fnforclass2)
precisionforclass2 = (tpforclass2) / (tpforclass2 + fpforclass2)
f1scoreforclass2 = (2 * precisionforclass2 * recallforclass2) / (precisionforclass2

tpforclass3 = cm[2][2]
fnforclass3 = cm[1][0] + cm[1][2]
tnforclass3 = cm[0][0] + cm[0][2] + cm[2][0] + cm[2][2]
fpforclass3 = cm[0][1] + cm[2][1]

recallforclass3 = (tpforclass3) / (tpforclass3 + fnforclass3)
precisionforclass3 = (tpforclass3) / (tpforclass3 + fpforclass3)
f1scoreforclass3 = (2 * precisionforclass3 * recallforclass3) / (precisionforclass3

print("Values of Class 1 evaluation metrics ---------- \n")
print("Precision of Class 1 is ")
print(precisionforclass1)
print("Recall of Class 1 is ")
print(recallforclass1)
```

```
print("F1-Score of Class 1 is ")
print(f1scoreforclass1)
print("Values of Class 2 evaluation metrics ---------- \n")
print("Precision of Class 2 is ")
print(precisionforclass2)
print("Recall of Class 2 is ")
print(recallforclass2)
print("F1-Score of Class 2 is ")
print(f1scoreforclass2)
print("Values of Class 3 evaluation metrics ---------- \n")
print("Precision of Class 3 is ")
print(precisionforclass3)
print("Recall of Class 3 is ")
print(recallforclass3)
print("F1-Score of Class 3 is ")
print(f1scoreforclass3)
```

```
Values of Class 1 evaluation metrics ----------

Precision of Class 1 is
0.776
Recall of Class 1 is
0.8584070796460177
F1-Score of Class 1 is
0.8151260504201682
Values of Class 2 evaluation metrics ----------

Precision of Class 2 is
0.6435643564356436
Recall of Class 2 is
0.625
F1-Score of Class 2 is
0.6341463414634146
Values of Class 3 evaluation metrics ----------

Precision of Class 3 is
0.8134715025906736
Recall of Class 3 is
0.8010204081632653
F1-Score of Class 3 is
0.80719794344473
```

2.2 Insights drawn (plots, markdown explanations)

1) We have used validation data in order to find the correct value of the hyperparameter Depth of the Decision Tree.
2) From the train accuracy and cross validation accuracy plots we can see that as the depth value increases, the train accuracy gets higher while cross-validation accuracy remains almost same indicating the occurence of overfitting with the increase in tree depth.
3) At Depth = 5, there is a proper balnace between Train accuracy and Cross-Validation accuracy which proves that it is the best value using which the performance of the model can be evaluated on unseen data.

3) The confusion matrix has very high values along the major diagonal indicating that the model is able to classify a very high number of unseen datapoints accurately.
4) The F1-Score for all the classes are > 0.5 indicating an appreciable performance of the model on unseen dataset.

3. Adaboost

3.1 Implementation of the Model

In [155…

```
import numpy as np
from numpy.core.umath_tests import inner1d
from copy import deepcopy
from sklearn.tree import DecisionTreeClassifier

class AdaBoostClassifier(object):


    def __init__(self, *args, **kwargs):
        if kwargs and args:
            raise ValueError(
                '''AdaBoostClassifier can only be called with keyword
                   arguments for the following keywords: base_estimator ,n_estimato
                    learning_rate,algorithm,random_state''')
        allowed_keys = ['base_estimator', 'n_estimators', 'learning_rate', 'algorit
        keywords_used = kwargs.keys()
        for keyword in keywords_used:
            if keyword not in allowed_keys:
                raise ValueError(keyword + ":  Wrong keyword used --- check spellin

        n_estimators = 50
        learning_rate = 1
        algorithm = 'SAMME.R'
        random_state = None

        if kwargs and not args:
            if 'base_estimator' in kwargs:
                base_estimator = kwargs.pop('base_estimator')
            else:
                raise ValueError('''base_estimator can not be None''')
            if 'n_estimators' in kwargs: n_estimators = kwargs.pop('n_estimators')
            if 'learning_rate' in kwargs: learning_rate = kwargs.pop('learning_rate
            if 'algorithm' in kwargs: algorithm = kwargs.pop('algorithm')
            if 'random_state' in kwargs: random_state = kwargs.pop('random_state')

        self.base_estimator_ = base_estimator
        self.n_estimators_ = n_estimators
        self.learning_rate_ = learning_rate
        self.algorithm_ = algorithm
        self.random_state_ = random_state
        self.estimators_ = list()
        self.estimator_weights_ = np.zeros(self.n_estimators_)
```

```
        self.estimator_errors_ = np.ones(self.n_estimators_)


    def _samme_proba(self, estimator, n_classes, X):

        proba = estimator.predict_proba(X)

        # Displace zero probabilities so the log is defined.
        # Also fix negative elements which may occur with
        # negative sample weights.
        proba[proba < np.finfo(proba.dtype).eps] = np.finfo(proba.dtype).eps
        log_proba = np.log(proba)

        return (n_classes - 1) * (log_proba - (1. / n_classes)
                                  * log_proba.sum(axis=1)[:, np.newaxis])


    def fit(self, X, y):
        self.n_samples = X.shape[0]
        # There is hidden trouble for classes, here the classes will be sorted.
        # So in boost we have to ensure that the predict results have the same clas
        self.classes_ = np.array(sorted(list(set(y))))
        self.n_classes_ = len(self.classes_)
        for iboost in range(self.n_estimators_):
            if iboost == 0:
                sample_weight = np.ones(self.n_samples) / self.n_samples

            sample_weight, estimator_weight, estimator_error = self.boost(X, y, sam

            # early stop
            if estimator_error == None:
                break

            # append error and weight
            self.estimator_errors_[iboost] = estimator_error
            self.estimator_weights_[iboost] = estimator_weight

            if estimator_error <= 0:
                break

        return self


    def boost(self, X, y, sample_weight):
        if self.algorithm_ == 'SAMME':
            return self.discrete_boost(X, y, sample_weight)
        elif self.algorithm_ == 'SAMME.R':
            return self.real_boost(X, y, sample_weight)

    def real_boost(self, X, y, sample_weight):
        estimator = deepcopy(self.base_estimator_)
        if self.random_state_:
            estimator.set_params(random_state=1)

        estimator.fit(X, y, sample_weight=sample_weight)
```

```
        y_pred = estimator.predict(X)
        incorrect = y_pred != y
        estimator_error = np.dot(incorrect, sample_weight) / np.sum(sample_weight,

        # if worse than random guess, stop boosting
        if estimator_error >= 1.0 - 1 / self.n_classes_:
            return None, None, None

        y_predict_proba = estimator.predict_proba(X)
        # repalce zero
        y_predict_proba[y_predict_proba < np.finfo(y_predict_proba.dtype).eps] = np

        y_codes = np.array([-1. / (self.n_classes_ - 1), 1.])
        y_coding = y_codes.take(np.array(self.classes_) == y[:, np.newaxis])

        # for sample weight update
        intermediate_variable = (-1. * self.learning_rate_ * (((self.n_classes_ - 1
                                                                inner1d(y_coding, np.
                                                                    y_predict_proba))

        # update sample weight
        sample_weight *= np.exp(intermediate_variable)

        sample_weight_sum = np.sum(sample_weight, axis=0)
        if sample_weight_sum <= 0:
            return None, None, None

        # normalize sample weight
        sample_weight /= sample_weight_sum

        # append the estimator
        self.estimators_.append(estimator)

        return sample_weight, 1, estimator_error


    def discrete_boost(self, X, y, sample_weight):
        estimator = deepcopy(self.base_estimator_)
        if self.random_state_:
            estimator.set_params(random_state=1)

        estimator.fit(X, y, sample_weight=sample_weight)

        y_pred = estimator.predict(X)
        incorrect = y_pred != y
        estimator_error = np.dot(incorrect, sample_weight) / np.sum(sample_weight,

        # if worse than random guess, stop boosting
        if estimator_error >= 1 - 1 / self.n_classes_:
            return None, None, None

        # update estimator_weight
        estimator_weight = self.learning_rate_ * np.log((1 - estimator_error) / est
            self.n_classes_ - 1)

        if estimator_weight <= 0:
```

```
            return None, None, None

        # update sample weight
        sample_weight *= np.exp(estimator_weight * incorrect)

        sample_weight_sum = np.sum(sample_weight, axis=0)
        if sample_weight_sum <= 0:
            return None, None, None

        # normalize sample weight
        sample_weight /= sample_weight_sum

        # append the estimator
        self.estimators_.append(estimator)

        return sample_weight, estimator_weight, estimator_error

    def predict(self, X):
        n_classes = self.n_classes_
        classes = self.classes_[:, np.newaxis]
        pred = None

        if self.algorithm_ == 'SAMME.R':
            # The weights are all 1. for SAMME.R
            pred = sum(self._samme_proba(estimator, n_classes, X) for estimator in
        else:  # self.algorithm == "SAMME"
            pred = sum((estimator.predict(X) == classes).T * w
                       for estimator, w in zip(self.estimators_,
                                               self.estimator_weights_))

        pred /= self.estimator_weights_.sum()
        if n_classes == 2:
            pred[:, 0] *= -1
            pred = pred.sum(axis=1)
            return self.classes_.take(pred > 0, axis=0)

        return self.classes_.take(np.argmax(pred, axis=1), axis=0)


    def predict_proba(self, X):
        if self.algorithm_ == 'SAMME.R':
            # The weights are all 1. for SAMME.R
            proba = sum(self._samme_proba(estimator, self.n_classes_, X)
                        for estimator in self.estimators_)
        else:  # self.algorithm == "SAMME"
            proba = sum(estimator.predict_proba(X) * w
                        for estimator, w in zip(self.estimators_,
                                                self.estimator_weights_))

        proba /= self.estimator_weights_.sum()
        proba = np.exp((1. / (n_classes - 1)) * proba)
        normalizer = proba.sum(axis=1)[:, np.newaxis]
        normalizer[normalizer == 0.0] = 1.0
        proba /= normalizer

        return proba
```

In [156…

```
train_accuracy_score=[]
val_accuracy_score=[]
learning_rate = []
num_estimators = []
for lr in np.arange(0.05,1.05,0.05):
    for ne in [10,50,100,500,1000]:

        bdt_real_test = AdaBoostClassifier(
            base_estimator=DecisionTreeClassifier(max_depth=3),
            n_estimators=ne,
            learning_rate=lr)
        bdt_real_test.fit(train_dataframe.iloc[:,0:14].to_numpy(), train_dataframe[
        pred = bdt_real_test.predict(validation_dataframe.iloc[:,0:14].to_numpy())
        val_acc = accuracy_score(pred,validation_dataframe['label'].to_numpy())
        train_pred = bdt_real_test.predict(train_dataframe.iloc[:,0:14].to_numpy())
        train_acc = accuracy_score(train_pred,train_dataframe['label'].to_numpy())
        train_accuracy_score.append(train_acc)
        val_accuracy_score.append(val_acc)
        learning_rate.append(lr)
        num_estimators.append(ne)
        print("\n learning rate = {}: ".format(lr), end = "")
        print(" Number of estimators = {}: ".format(ne), end = "")
        print("accuracy of CV = {0:.2f}%, ".format(val_acc), end = "")
        print("accuracy of Train = {0:.2f}%, ".format(train_acc), end = "")
```

```
 learning rate = 0.05:  Number of estimators = 10: accuracy of CV = 0.65%, accuracy
of Train = 0.70%,
 learning rate = 0.05:  Number of estimators = 50: accuracy of CV = 0.65%, accuracy
of Train = 0.78%,
 learning rate = 0.05:  Number of estimators = 100: accuracy of CV = 0.62%, accuracy
of Train = 0.83%,
 learning rate = 0.05:  Number of estimators = 500: accuracy of CV = 0.62%, accuracy
of Train = 0.92%,
 learning rate = 0.05:  Number of estimators = 1000: accuracy of CV = 0.59%, accurac
y of Train = 0.95%,
 learning rate = 0.1:  Number of estimators = 10: accuracy of CV = 0.68%, accuracy o
f Train = 0.71%,
 learning rate = 0.1:  Number of estimators = 50: accuracy of CV = 0.66%, accuracy o
f Train = 0.81%,
 learning rate = 0.1:  Number of estimators = 100: accuracy of CV = 0.65%, accuracy
of Train = 0.87%,
 learning rate = 0.1:  Number of estimators = 500: accuracy of CV = 0.61%, accuracy
of Train = 0.95%,
 learning rate = 0.1:  Number of estimators = 1000: accuracy of CV = 0.63%, accuracy
of Train = 0.96%,
 learning rate = 0.15000000000000002:  Number of estimators = 10: accuracy of CV =
0.67%, accuracy of Train = 0.73%,
 learning rate = 0.15000000000000002:  Number of estimators = 50: accuracy of CV =
0.64%, accuracy of Train = 0.85%,
 learning rate = 0.15000000000000002:  Number of estimators = 100: accuracy of CV =
0.61%, accuracy of Train = 0.88%,
 learning rate = 0.15000000000000002:  Number of estimators = 500: accuracy of CV =
0.60%, accuracy of Train = 0.96%,
 learning rate = 0.15000000000000002:  Number of estimators = 1000: accuracy of CV =
0.62%, accuracy of Train = 0.97%,
 learning rate = 0.2:  Number of estimators = 10: accuracy of CV = 0.65%, accuracy o
```

f Train = 0.75%,
 learning rate = 0.2: Number of estimators = 50: accuracy of CV = 0.60%, accuracy o
f Train = 0.84%,
 learning rate = 0.2: Number of estimators = 100: accuracy of CV = 0.60%, accuracy
of Train = 0.89%,
 learning rate = 0.2: Number of estimators = 500: accuracy of CV = 0.62%, accuracy
of Train = 0.96%,
 learning rate = 0.2: Number of estimators = 1000: accuracy of CV = 0.65%, accuracy
of Train = 0.98%,
 learning rate = 0.25: Number of estimators = 10: accuracy of CV = 0.65%, accuracy
of Train = 0.73%,
 learning rate = 0.25: Number of estimators = 50: accuracy of CV = 0.61%, accuracy
of Train = 0.83%,
 learning rate = 0.25: Number of estimators = 100: accuracy of CV = 0.57%, accuracy
of Train = 0.89%,
 learning rate = 0.25: Number of estimators = 500: accuracy of CV = 0.62%, accuracy
of Train = 0.96%,
 learning rate = 0.25: Number of estimators = 1000: accuracy of CV = 0.63%, accurac
y of Train = 0.98%,
 learning rate = 0.3: Number of estimators = 10: accuracy of CV = 0.64%, accuracy o
f Train = 0.75%,
 learning rate = 0.3: Number of estimators = 50: accuracy of CV = 0.61%, accuracy o
f Train = 0.85%,
 learning rate = 0.3: Number of estimators = 100: accuracy of CV = 0.58%, accuracy
of Train = 0.91%,
 learning rate = 0.3: Number of estimators = 500: accuracy of CV = 0.63%, accuracy
of Train = 0.97%,
 learning rate = 0.3: Number of estimators = 1000: accuracy of CV = 0.63%, accuracy
of Train = 0.98%,
 learning rate = 0.35000000000000003: Number of estimators = 10: accuracy of CV =
0.67%, accuracy of Train = 0.76%,
 learning rate = 0.35000000000000003: Number of estimators = 50: accuracy of CV =
0.58%, accuracy of Train = 0.88%,
 learning rate = 0.35000000000000003: Number of estimators = 100: accuracy of CV =
0.59%, accuracy of Train = 0.90%,
 learning rate = 0.35000000000000003: Number of estimators = 500: accuracy of CV =
0.62%, accuracy of Train = 0.97%,
 learning rate = 0.35000000000000003: Number of estimators = 1000: accuracy of CV =
0.61%, accuracy of Train = 0.99%,
 learning rate = 0.4: Number of estimators = 10: accuracy of CV = 0.65%, accuracy o
f Train = 0.75%,
 learning rate = 0.4: Number of estimators = 50: accuracy of CV = 0.60%, accuracy o
f Train = 0.86%,
 learning rate = 0.4: Number of estimators = 100: accuracy of CV = 0.56%, accuracy
of Train = 0.91%,
 learning rate = 0.4: Number of estimators = 500: accuracy of CV = 0.62%, accuracy
of Train = 0.98%,
 learning rate = 0.4: Number of estimators = 1000: accuracy of CV = 0.61%, accuracy
of Train = 0.99%,
 learning rate = 0.45: Number of estimators = 10: accuracy of CV = 0.65%, accuracy
of Train = 0.76%,
 learning rate = 0.45: Number of estimators = 50: accuracy of CV = 0.58%, accuracy
of Train = 0.84%,
 learning rate = 0.45: Number of estimators = 100: accuracy of CV = 0.59%, accuracy
of Train = 0.93%,
 learning rate = 0.45: Number of estimators = 500: accuracy of CV = 0.62%, accuracy

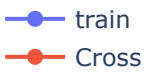
of Train = 0.97%,
 learning rate = 0.45: Number of estimators = 1000: accuracy of CV = 0.63%, accurac
y of Train = 0.98%,
 learning rate = 0.5: Number of estimators = 10: accuracy of CV = 0.64%, accuracy o
f Train = 0.75%,
 learning rate = 0.5: Number of estimators = 50: accuracy of CV = 0.61%, accuracy o
f Train = 0.87%,
 learning rate = 0.5: Number of estimators = 100: accuracy of CV = 0.62%, accuracy
of Train = 0.91%,
 learning rate = 0.5: Number of estimators = 500: accuracy of CV = 0.62%, accuracy
of Train = 0.98%,
 learning rate = 0.5: Number of estimators = 1000: accuracy of CV = 0.63%, accuracy
of Train = 0.98%,
 learning rate = 0.55: Number of estimators = 10: accuracy of CV = 0.63%, accuracy
of Train = 0.76%,
 learning rate = 0.55: Number of estimators = 50: accuracy of CV = 0.60%, accuracy
of Train = 0.87%,
 learning rate = 0.55: Number of estimators = 100: accuracy of CV = 0.61%, accuracy
of Train = 0.93%,
 learning rate = 0.55: Number of estimators = 500: accuracy of CV = 0.63%, accuracy
of Train = 0.98%,
 learning rate = 0.55: Number of estimators = 1000: accuracy of CV = 0.62%, accurac
y of Train = 0.98%,
 learning rate = 0.6000000000000001: Number of estimators = 10: accuracy of CV = 0.
66%, accuracy of Train = 0.78%,
 learning rate = 0.6000000000000001: Number of estimators = 50: accuracy of CV = 0.
62%, accuracy of Train = 0.87%,
 learning rate = 0.6000000000000001: Number of estimators = 100: accuracy of CV =
0.60%, accuracy of Train = 0.92%,
 learning rate = 0.6000000000000001: Number of estimators = 500: accuracy of CV =
0.65%, accuracy of Train = 0.97%,
 learning rate = 0.6000000000000001: Number of estimators = 1000: accuracy of CV =
0.66%, accuracy of Train = 0.99%,
 learning rate = 0.6500000000000001: Number of estimators = 10: accuracy of CV = 0.
67%, accuracy of Train = 0.76%,
 learning rate = 0.6500000000000001: Number of estimators = 50: accuracy of CV = 0.
61%, accuracy of Train = 0.88%,
 learning rate = 0.6500000000000001: Number of estimators = 100: accuracy of CV =
0.58%, accuracy of Train = 0.93%,
 learning rate = 0.6500000000000001: Number of estimators = 500: accuracy of CV =
0.62%, accuracy of Train = 0.98%,
 learning rate = 0.6500000000000001: Number of estimators = 1000: accuracy of CV =
0.63%, accuracy of Train = 0.99%,
 learning rate = 0.7000000000000001: Number of estimators = 10: accuracy of CV = 0.
65%, accuracy of Train = 0.75%,
 learning rate = 0.7000000000000001: Number of estimators = 50: accuracy of CV = 0.
62%, accuracy of Train = 0.88%,
 learning rate = 0.7000000000000001: Number of estimators = 100: accuracy of CV =
0.60%, accuracy of Train = 0.91%,
 learning rate = 0.7000000000000001: Number of estimators = 500: accuracy of CV =
0.65%, accuracy of Train = 0.98%,
 learning rate = 0.7000000000000001: Number of estimators = 1000: accuracy of CV =
0.63%, accuracy of Train = 0.98%,
 learning rate = 0.7500000000000001: Number of estimators = 10: accuracy of CV = 0.
64%, accuracy of Train = 0.77%,
 learning rate = 0.7500000000000001: Number of estimators = 50: accuracy of CV = 0.

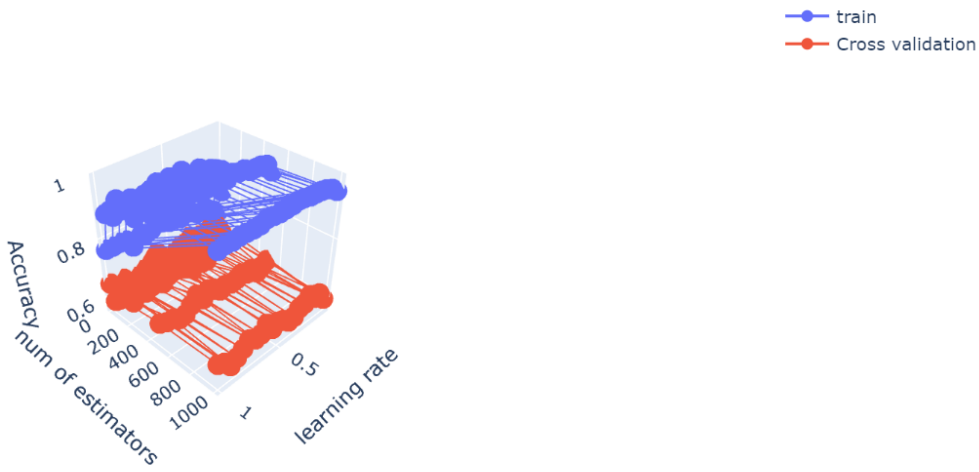
60%, accuracy of Train = 0.87%,
 learning rate = 0.7500000000000001: Number of estimators = 100: accuracy of CV =
0.60%, accuracy of Train = 0.91%,
 learning rate = 0.7500000000000001: Number of estimators = 500: accuracy of CV =
0.63%, accuracy of Train = 0.98%,
 learning rate = 0.7500000000000001: Number of estimators = 1000: accuracy of CV =
0.65%, accuracy of Train = 0.98%,
 learning rate = 0.8: Number of estimators = 10: accuracy of CV = 0.62%, accuracy o
f Train = 0.73%,
 learning rate = 0.8: Number of estimators = 50: accuracy of CV = 0.60%, accuracy o
f Train = 0.84%,
 learning rate = 0.8: Number of estimators = 100: accuracy of CV = 0.59%, accuracy
of Train = 0.89%,
 learning rate = 0.8: Number of estimators = 500: accuracy of CV = 0.61%, accuracy
of Train = 0.98%,
 learning rate = 0.8: Number of estimators = 1000: accuracy of CV = 0.63%, accuracy
of Train = 0.98%,
 learning rate = 0.8500000000000001: Number of estimators = 10: accuracy of CV = 0.
60%, accuracy of Train = 0.76%,
 learning rate = 0.8500000000000001: Number of estimators = 50: accuracy of CV = 0.
61%, accuracy of Train = 0.89%,
 learning rate = 0.8500000000000001: Number of estimators = 100: accuracy of CV =
0.60%, accuracy of Train = 0.92%,
 learning rate = 0.8500000000000001: Number of estimators = 500: accuracy of CV =
0.61%, accuracy of Train = 0.97%,
 learning rate = 0.8500000000000001: Number of estimators = 1000: accuracy of CV =
0.61%, accuracy of Train = 0.98%,
 learning rate = 0.9000000000000001: Number of estimators = 10: accuracy of CV = 0.
64%, accuracy of Train = 0.77%,
 learning rate = 0.9000000000000001: Number of estimators = 50: accuracy of CV = 0.
61%, accuracy of Train = 0.88%,
 learning rate = 0.9000000000000001: Number of estimators = 100: accuracy of CV =
0.58%, accuracy of Train = 0.90%,
 learning rate = 0.9000000000000001: Number of estimators = 500: accuracy of CV =
0.62%, accuracy of Train = 0.97%,
 learning rate = 0.9000000000000001: Number of estimators = 1000: accuracy of CV =
0.60%, accuracy of Train = 0.98%,
 learning rate = 0.9500000000000001: Number of estimators = 10: accuracy of CV = 0.
62%, accuracy of Train = 0.77%,
 learning rate = 0.9500000000000001: Number of estimators = 50: accuracy of CV = 0.
60%, accuracy of Train = 0.87%,
 learning rate = 0.9500000000000001: Number of estimators = 100: accuracy of CV =
0.62%, accuracy of Train = 0.93%,
 learning rate = 0.9500000000000001: Number of estimators = 500: accuracy of CV =
0.62%, accuracy of Train = 0.97%,
 learning rate = 0.9500000000000001: Number of estimators = 1000: accuracy of CV =
0.63%, accuracy of Train = 0.98%,
 learning rate = 1.0: Number of estimators = 10: accuracy of CV = 0.65%, accuracy o
f Train = 0.77%,
 learning rate = 1.0: Number of estimators = 50: accuracy of CV = 0.60%, accuracy o
f Train = 0.89%,
 learning rate = 1.0: Number of estimators = 100: accuracy of CV = 0.61%, accuracy
of Train = 0.91%,
 learning rate = 1.0: Number of estimators = 500: accuracy of CV = 0.63%, accuracy
of Train = 0.97%,

```
 learning rate = 1.0:  Number of estimators = 1000: accuracy of CV = 0.64%, accuracy
of Train = 0.98%,
```

In [157…

```
trace1 = go.Scatter3d(x=learning_rate,y=num_estimators,z= train_accuracy_score, nam
trace2 = go.Scatter3d(x=learning_rate,y=num_estimators,z=val_accuracy_score, name =
data = [trace1, trace2]

layout = go.Layout(scene = dict(
        xaxis = dict(title='learning rate'),
        yaxis = dict(title='num of estimators'),
        zaxis = dict(title='Accuracy'),))

fig = go.Figure(data=data, layout=layout)
offline.iplot(fig, filename='3d-scatter-colorscale')
```

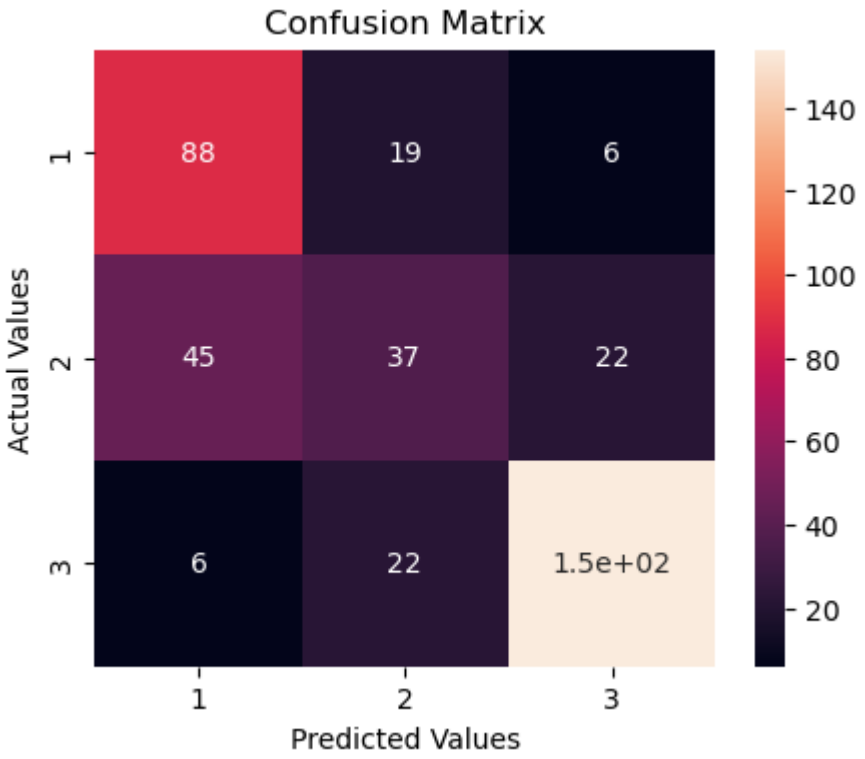




In [158…
```
print('Best num of estimators is',10,'Best learning rate ',0.4)
#print(accuracy_score(bdt_real.predict(X_test),y_test))
bdt_real_test = AdaBoostClassifier(
            base_estimator=DecisionTreeClassifier(max_depth=3),
            n_estimators=10,
            learning_rate=0.4)
bdt_real_test.fit(train_dataframe.iloc[:,0:14].to_numpy(), train_dataframe['label']
test_pred = bdt_real_test.predict(test_dataframe.iloc[:,0:14].to_numpy())
print(accuracy_score(test_pred,test_dataframe['label'].to_numpy()))
```

```
Best num of estimators is 10 Best learning rate  0.4
0.6992481203007519
```

In [159…
```
cm = confusion_matrix(y_test, test_pred)
cm_df = pd.DataFrame(cm,index = ['1','2','3'], columns = ['1','2','3'])
plt.figure(figsize=(5,4))
sns.heatmap(cm_df, annot=True)
plt.title('Confusion Matrix')
plt.ylabel('Actual Values')
plt.xlabel('Predicted Values')
plt.show()
```



In [160…

```
tpforclass1 = cm[0][0]
fnforclass1 = cm[0][1] + cm[0][2]
tnforclass1 = cm[1][1] + cm[1][2] + cm[2][1] + cm[2][2]
fpforclass1 = cm[1][0] + cm[2][0]

recallforclass1 = (tpforclass1) / (tpforclass1 + fnforclass1)
precisionforclass1 = (tpforclass1) / (tpforclass1 + fpforclass1)
f1scoreforclass1 = (2 * precisionforclass1 * recallforclass1) / (precisionforclass1

tpforclass2 = cm[1][1]
fnforclass2 = cm[1][0] + cm[1][2]
tnforclass2 = cm[0][0] + cm[0][2] + cm[2][0] + cm[2][2]
fpforclass2 = cm[0][1] + cm[2][1]

recallforclass2 = (tpforclass2) / (tpforclass2 + fnforclass2)
precisionforclass2 = (tpforclass2) / (tpforclass2 + fpforclass2)
f1scoreforclass2 = (2 * precisionforclass2 * recallforclass2) / (precisionforclass2

tpforclass3 = cm[2][2]
fnforclass3 = cm[1][0] + cm[1][2]
tnforclass3 = cm[0][0] + cm[0][2] + cm[2][0] + cm[2][2]
fpforclass3 = cm[0][1] + cm[2][1]

recallforclass3 = (tpforclass3) / (tpforclass3 + fnforclass3)
precisionforclass3 = (tpforclass3) / (tpforclass3 + fpforclass3)
f1scoreforclass3 = (2 * precisionforclass3 * recallforclass3) / (precisionforclass3

print("Values of Class 1 evaluation metrics ---------- \n")
print("Precision of Class 1 is ")
print(precisionforclass1)
print("Recall of Class 1 is ")
print(recallforclass1)
```

```
print("F1-Score of Class 1 is ")
print(f1scoreforclass1)
print("Values of Class 2 evaluation metrics ---------- \n")
print("Precision of Class 2 is ")
print(precisionforclass2)
print("Recall of Class 2 is ")
print(recallforclass2)
print("F1-Score of Class 2 is ")
print(f1scoreforclass2)
print("Values of Class 3 evaluation metrics ---------- \n")
print("Precision of Class 3 is ")
print(precisionforclass3)
print("Recall of Class 3 is ")
print(recallforclass3)
print("F1-Score of Class 3 is ")
print(f1scoreforclass3)
```

```
Values of Class 1 evaluation metrics ----------

Precision of Class 1 is
0.6330935251798561
Recall of Class 1 is
0.7787610619469026
F1-Score of Class 1 is
0.6984126984126984
Values of Class 2 evaluation metrics ----------

Precision of Class 2 is
0.47435897435897434
Recall of Class 2 is
0.3557692307692308
F1-Score of Class 2 is
0.4065934065934066
Values of Class 3 evaluation metrics ----------

Precision of Class 3 is
0.7897435897435897
Recall of Class 3 is
0.6968325791855203
F1-Score of Class 3 is
0.7403846153846153
```

3.2 Insights drawn (plots, markdown explanations)

1) Here we have taken the Base Learners to be Decision Trees. 1) We have used validation data in order to find the correct value of two hyperparameters: number of estimators and learning rate.

2) From the train accuracy and cross validation accuracy plots we can see that as the number of base learners and learning rate increases, the train accuracy gets higher while cross-validation accuracy remains almost same indicating the occurence of overfitting with the increase in the number of base learners and learning rate.

3) At Number of estimators = 10 and learning rate = 0.4 , there is a proper balance between

Train accuracy and Cross-Validation accuracy which proves that it is the best value using which the performance of the model can be evaluated on unseen data.
4) The confusion matrix has very high values along the major diagonal indicating that the model is able to classify a very high number of unseen datapoints accurately.
5) The F1-Score for all the classes are > 0.5 indicating an appreciable performance of the model on unseen dataset.

4. Multiclass SVM

4.1 Implementation of the Model

In [161…

```
import numpy as np

class MulticlassSVM:
    def __init__(self,learning_rate=0.01, epochs=1000, regularization_strength=1):
        self.learning_rate = learning_rate
        self.epochs = epochs
        self.regularization_strength = regularization_strength
        self.weights = None
        self.classes = None

    def one_vs_all(self, X, y, class_label):
        binary_labels = np.where(y == class_label, 1, -1)
        weights = np.zeros(X.shape[1])
        for epoch in range(self.epochs):
            for i, x in enumerate(X):
                margin = binary_labels[i] * np.dot(weights, x)
                if margin < 1:
                    gradient = -binary_labels[i] * x + 2 * self.regularization_stre
                else:
                    gradient = 2 * self.regularization_strength * weights
                weights -= self.learning_rate * gradient
        return weights

    def fit(self, X, y):
        self.classes = np.unique(y)
        self.weights = np.zeros((len(self.classes), X.shape[1]))

        for i, class_label in enumerate(self.classes):
            self.weights[i, :] = self.one_vs_all(X, y, class_label)

    def predict_one_vs_all(self, x):
        scores = np.dot(self.weights, x)
        return np.argmax(scores)

    def predict(self, X):
        predictions = [self.predict_one_vs_all(x) for x in X]
        return np.array(predictions)

# Example usage:
```

```
# Assume X_train and y_train are your training data
# Assume X_test is your test data
```

In [162…
```
svm = MulticlassSVM()
svm.fit(X_train, y_train)
predictions = svm.predict(X_test)
```

In [163…
```
train_accuracy_score=[]
val_accuracy_score=[]
learning_rate = []
regularization_strength = []
for lr in np.arange(0.05,1.05,0.05):
    for rs in [0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1]:

        svm = MulticlassSVM(learning_rate = lr,regularization_strength = rs)
        svm.fit(X_train, y_train)
        pred = svm.predict(X_validation)
        val_acc = accuracy_score(pred,y_validation)
        train_pred = svm.predict(X_train)
        train_acc = accuracy_score(train_pred,y_train)
        train_accuracy_score.append(train_acc)
        val_accuracy_score.append(val_acc)
        learning_rate.append(lr)
        regularization_strength.append(rs)
        print("\n learning rate = {}: ".format(lr), end = "")
        print(" Regularisation Strength = {}: ".format(rs), end = "")
        print("accuracy of CV = {0:.2f}%, ".format(val_acc), end = "")
        print("accuracy of Train = {0:.2f}%, ".format(train_acc), end = "")
```

```
 learning rate = 0.05:  Regularisation Strength = 0.2: accuracy of CV = 0.13%, accur
acy of Train = 0.12%,
 learning rate = 0.05:  Regularisation Strength = 0.3: accuracy of CV = 0.15%, accur
acy of Train = 0.14%,
 learning rate = 0.05:  Regularisation Strength = 0.4: accuracy of CV = 0.16%, accur
acy of Train = 0.14%,
 learning rate = 0.05:  Regularisation Strength = 0.5: accuracy of CV = 0.17%, accur
acy of Train = 0.14%,
 learning rate = 0.05:  Regularisation Strength = 0.6: accuracy of CV = 0.18%, accur
acy of Train = 0.14%,
 learning rate = 0.05:  Regularisation Strength = 0.7: accuracy of CV = 0.18%, accur
acy of Train = 0.14%,
 learning rate = 0.05:  Regularisation Strength = 0.8: accuracy of CV = 0.18%, accur
acy of Train = 0.14%,
 learning rate = 0.05:  Regularisation Strength = 0.9: accuracy of CV = 0.18%, accur
acy of Train = 0.15%,
 learning rate = 0.05:  Regularisation Strength = 1: accuracy of CV = 0.17%, accurac
y of Train = 0.14%,
 learning rate = 0.1:  Regularisation Strength = 0.2: accuracy of CV = 0.20%, accura
cy of Train = 0.16%,
 learning rate = 0.1:  Regularisation Strength = 0.3: accuracy of CV = 0.19%, accura
cy of Train = 0.16%,
 learning rate = 0.1:  Regularisation Strength = 0.4: accuracy of CV = 0.20%, accura
cy of Train = 0.17%,
 learning rate = 0.1:  Regularisation Strength = 0.5: accuracy of CV = 0.17%, accura
cy of Train = 0.15%,
 learning rate = 0.1:  Regularisation Strength = 0.6: accuracy of CV = 0.18%, accura
```

cy of Train = 0.15%,
 learning rate = 0.1: Regularisation Strength = 0.7: accuracy of CV = 0.16%, accura
cy of Train = 0.14%,
 learning rate = 0.1: Regularisation Strength = 0.8: accuracy of CV = 0.18%, accura
cy of Train = 0.14%,
 learning rate = 0.1: Regularisation Strength = 0.9: accuracy of CV = 0.18%, accura
cy of Train = 0.14%,
 learning rate = 0.1: Regularisation Strength = 1: accuracy of CV = 0.18%, accuracy
of Train = 0.14%,
 learning rate = 0.15000000000000002: Regularisation Strength = 0.2: accuracy of CV
= 0.20%, accuracy of Train = 0.18%,
 learning rate = 0.15000000000000002: Regularisation Strength = 0.3: accuracy of CV
= 0.20%, accuracy of Train = 0.17%,
 learning rate = 0.15000000000000002: Regularisation Strength = 0.4: accuracy of CV
= 0.19%, accuracy of Train = 0.17%,
 learning rate = 0.15000000000000002: Regularisation Strength = 0.5: accuracy of CV
= 0.18%, accuracy of Train = 0.15%,
 learning rate = 0.15000000000000002: Regularisation Strength = 0.6: accuracy of CV
= 0.16%, accuracy of Train = 0.14%,
 learning rate = 0.15000000000000002: Regularisation Strength = 0.7: accuracy of CV
= 0.16%, accuracy of Train = 0.14%,
 learning rate = 0.15000000000000002: Regularisation Strength = 0.8: accuracy of CV
= 0.16%, accuracy of Train = 0.13%,
 learning rate = 0.15000000000000002: Regularisation Strength = 0.9: accuracy of CV
= 0.16%, accuracy of Train = 0.13%,
 learning rate = 0.15000000000000002: Regularisation Strength = 1: accuracy of CV =
0.16%, accuracy of Train = 0.12%,
 learning rate = 0.2: Regularisation Strength = 0.2: accuracy of CV = 0.19%, accura
cy of Train = 0.17%,
 learning rate = 0.2: Regularisation Strength = 0.3: accuracy of CV = 0.20%, accura
cy of Train = 0.17%,
 learning rate = 0.2: Regularisation Strength = 0.4: accuracy of CV = 0.20%, accura
cy of Train = 0.17%,
 learning rate = 0.2: Regularisation Strength = 0.5: accuracy of CV = 0.18%, accura
cy of Train = 0.14%,
 learning rate = 0.2: Regularisation Strength = 0.6: accuracy of CV = 0.17%, accura
cy of Train = 0.13%,
 learning rate = 0.2: Regularisation Strength = 0.7: accuracy of CV = 0.16%, accura
cy of Train = 0.13%,
 learning rate = 0.2: Regularisation Strength = 0.8: accuracy of CV = 0.17%, accura
cy of Train = 0.13%,
 learning rate = 0.2: Regularisation Strength = 0.9: accuracy of CV = 0.16%, accura
cy of Train = 0.13%,
 learning rate = 0.2: Regularisation Strength = 1: accuracy of CV = 0.17%, accuracy
of Train = 0.13%,
 learning rate = 0.25: Regularisation Strength = 0.2: accuracy of CV = 0.18%, accur
acy of Train = 0.17%,
 learning rate = 0.25: Regularisation Strength = 0.3: accuracy of CV = 0.20%, accur
acy of Train = 0.17%,
 learning rate = 0.25: Regularisation Strength = 0.4: accuracy of CV = 0.18%, accur
acy of Train = 0.15%,
 learning rate = 0.25: Regularisation Strength = 0.5: accuracy of CV = 0.17%, accur
acy of Train = 0.13%,
 learning rate = 0.25: Regularisation Strength = 0.6: accuracy of CV = 0.16%, accur
acy of Train = 0.13%,
 learning rate = 0.25: Regularisation Strength = 0.7: accuracy of CV = 0.17%, accur

acy of Train = 0.15%,
 learning rate = 0.25: Regularisation Strength = 0.8: accuracy of CV = 0.17%, accur
acy of Train = 0.14%,
 learning rate = 0.25: Regularisation Strength = 0.9: accuracy of CV = 0.17%, accur
acy of Train = 0.14%,
 learning rate = 0.25: Regularisation Strength = 1: accuracy of CV = 0.17%, accurac
y of Train = 0.15%,
 learning rate = 0.3: Regularisation Strength = 0.2: accuracy of CV = 0.21%, accura
cy of Train = 0.17%,
 learning rate = 0.3: Regularisation Strength = 0.3: accuracy of CV = 0.20%, accura
cy of Train = 0.17%,
 learning rate = 0.3: Regularisation Strength = 0.4: accuracy of CV = 0.18%, accura
cy of Train = 0.15%,
 learning rate = 0.3: Regularisation Strength = 0.5: accuracy of CV = 0.17%, accura
cy of Train = 0.14%,
 learning rate = 0.3: Regularisation Strength = 0.6: accuracy of CV = 0.17%, accura
cy of Train = 0.14%,
 learning rate = 0.3: Regularisation Strength = 0.7: accuracy of CV = 0.17%, accura
cy of Train = 0.14%,
 learning rate = 0.3: Regularisation Strength = 0.8: accuracy of CV = 0.17%, accura
cy of Train = 0.14%,
 learning rate = 0.3: Regularisation Strength = 0.9: accuracy of CV = 0.17%, accura
cy of Train = 0.15%,
 learning rate = 0.3: Regularisation Strength = 1: accuracy of CV = 0.18%, accuracy
of Train = 0.15%,
 learning rate = 0.35000000000000003: Regularisation Strength = 0.2: accuracy of CV
= 0.20%, accuracy of Train = 0.17%,
 learning rate = 0.35000000000000003: Regularisation Strength = 0.3: accuracy of CV
= 0.20%, accuracy of Train = 0.17%,
 learning rate = 0.35000000000000003: Regularisation Strength = 0.4: accuracy of CV
= 0.17%, accuracy of Train = 0.15%,
 learning rate = 0.35000000000000003: Regularisation Strength = 0.5: accuracy of CV
= 0.17%, accuracy of Train = 0.14%,
 learning rate = 0.35000000000000003: Regularisation Strength = 0.6: accuracy of CV
= 0.17%, accuracy of Train = 0.14%,
 learning rate = 0.35000000000000003: Regularisation Strength = 0.7: accuracy of CV
= 0.17%, accuracy of Train = 0.15%,
 learning rate = 0.35000000000000003: Regularisation Strength = 0.8: accuracy of CV
= 0.18%, accuracy of Train = 0.15%,
 learning rate = 0.35000000000000003: Regularisation Strength = 0.9: accuracy of CV
= 0.19%, accuracy of Train = 0.15%,
 learning rate = 0.35000000000000003: Regularisation Strength = 1: accuracy of CV =
0.19%, accuracy of Train = 0.17%,
 learning rate = 0.4: Regularisation Strength = 0.2: accuracy of CV = 0.19%, accura
cy of Train = 0.17%,
 learning rate = 0.4: Regularisation Strength = 0.3: accuracy of CV = 0.17%, accura
cy of Train = 0.14%,
 learning rate = 0.4: Regularisation Strength = 0.4: accuracy of CV = 0.16%, accura
cy of Train = 0.14%,
 learning rate = 0.4: Regularisation Strength = 0.5: accuracy of CV = 0.17%, accura
cy of Train = 0.14%,
 learning rate = 0.4: Regularisation Strength = 0.6: accuracy of CV = 0.18%, accura
cy of Train = 0.14%,
 learning rate = 0.4: Regularisation Strength = 0.7: accuracy of CV = 0.18%, accura
cy of Train = 0.15%,
 learning rate = 0.4: Regularisation Strength = 0.8: accuracy of CV = 0.19%, accura

cy of Train = 0.16%,
 learning rate = 0.4: Regularisation Strength = 0.9: accuracy of CV = 0.19%, accura
cy of Train = 0.17%,
 learning rate = 0.4: Regularisation Strength = 1: accuracy of CV = 0.19%, accuracy
of Train = 0.17%,
 learning rate = 0.45: Regularisation Strength = 0.2: accuracy of CV = 0.20%, accur
acy of Train = 0.20%,
 learning rate = 0.45: Regularisation Strength = 0.3: accuracy of CV = 0.17%, accur
acy of Train = 0.14%,
 learning rate = 0.45: Regularisation Strength = 0.4: accuracy of CV = 0.17%, accur
acy of Train = 0.14%,
 learning rate = 0.45: Regularisation Strength = 0.5: accuracy of CV = 0.17%, accur
acy of Train = 0.14%,
 learning rate = 0.45: Regularisation Strength = 0.6: accuracy of CV = 0.17%, accur
acy of Train = 0.15%,
 learning rate = 0.45: Regularisation Strength = 0.7: accuracy of CV = 0.19%, accur
acy of Train = 0.15%,
 learning rate = 0.45: Regularisation Strength = 0.8: accuracy of CV = 0.19%, accur
acy of Train = 0.17%,
 learning rate = 0.45: Regularisation Strength = 0.9: accuracy of CV = 0.19%, accur
acy of Train = 0.17%,
 learning rate = 0.45: Regularisation Strength = 1: accuracy of CV = 0.18%, accurac
y of Train = 0.17%,
 learning rate = 0.5: Regularisation Strength = 0.2: accuracy of CV = 0.20%, accura
cy of Train = 0.20%,
 learning rate = 0.5: Regularisation Strength = 0.3: accuracy of CV = 0.16%, accura
cy of Train = 0.14%,
 learning rate = 0.5: Regularisation Strength = 0.4: accuracy of CV = 0.17%, accura
cy of Train = 0.14%,
 learning rate = 0.5: Regularisation Strength = 0.5: accuracy of CV = 0.17%, accura
cy of Train = 0.14%,
 learning rate = 0.5: Regularisation Strength = 0.6: accuracy of CV = 0.18%, accura
cy of Train = 0.15%,
 learning rate = 0.5: Regularisation Strength = 0.7: accuracy of CV = 0.19%, accura
cy of Train = 0.17%,
 learning rate = 0.5: Regularisation Strength = 0.8: accuracy of CV = 0.19%, accura
cy of Train = 0.17%,
 learning rate = 0.5: Regularisation Strength = 0.9: accuracy of CV = 0.18%, accura
cy of Train = 0.17%,
 learning rate = 0.5: Regularisation Strength = 1: accuracy of CV = 0.13%, accuracy
of Train = 0.13%,
 learning rate = 0.55: Regularisation Strength = 0.2: accuracy of CV = 0.21%, accur
acy of Train = 0.20%,
 learning rate = 0.55: Regularisation Strength = 0.3: accuracy of CV = 0.17%, accur
acy of Train = 0.14%,
 learning rate = 0.55: Regularisation Strength = 0.4: accuracy of CV = 0.17%, accur
acy of Train = 0.14%,
 learning rate = 0.55: Regularisation Strength = 0.5: accuracy of CV = 0.18%, accur
acy of Train = 0.15%,
 learning rate = 0.55: Regularisation Strength = 0.6: accuracy of CV = 0.20%, accur
acy of Train = 0.16%,
 learning rate = 0.55: Regularisation Strength = 0.7: accuracy of CV = 0.19%, accur
acy of Train = 0.18%,
 learning rate = 0.55: Regularisation Strength = 0.8: accuracy of CV = 0.18%, accur
acy of Train = 0.17%,
 learning rate = 0.55: Regularisation Strength = 0.9: accuracy of CV = 0.17%, accur

acy of Train = 0.17%,
 learning rate = 0.55: Regularisation Strength = 1: accuracy of CV = 0.18%, accurac
y of Train = 0.20%,
 learning rate = 0.6000000000000001: Regularisation Strength = 0.2: accuracy of CV
= 0.21%, accuracy of Train = 0.17%,
 learning rate = 0.6000000000000001: Regularisation Strength = 0.3: accuracy of CV
= 0.17%, accuracy of Train = 0.14%,
 learning rate = 0.6000000000000001: Regularisation Strength = 0.4: accuracy of CV
= 0.18%, accuracy of Train = 0.14%,
 learning rate = 0.6000000000000001: Regularisation Strength = 0.5: accuracy of CV
= 0.18%, accuracy of Train = 0.15%,
 learning rate = 0.6000000000000001: Regularisation Strength = 0.6: accuracy of CV
= 0.19%, accuracy of Train = 0.17%,
 learning rate = 0.6000000000000001: Regularisation Strength = 0.7: accuracy of CV
= 0.19%, accuracy of Train = 0.17%,
 learning rate = 0.6000000000000001: Regularisation Strength = 0.8: accuracy of CV
= 0.17%, accuracy of Train = 0.17%,
 learning rate = 0.6000000000000001: Regularisation Strength = 0.9: accuracy of CV
= 0.18%, accuracy of Train = 0.20%,
 learning rate = 0.6000000000000001: Regularisation Strength = 1: accuracy of CV =
0.13%, accuracy of Train = 0.14%,
 learning rate = 0.6500000000000001: Regularisation Strength = 0.2: accuracy of CV
= 0.21%, accuracy of Train = 0.17%,
 learning rate = 0.6500000000000001: Regularisation Strength = 0.3: accuracy of CV
= 0.17%, accuracy of Train = 0.14%,
 learning rate = 0.6500000000000001: Regularisation Strength = 0.4: accuracy of CV
= 0.17%, accuracy of Train = 0.15%,
 learning rate = 0.6500000000000001: Regularisation Strength = 0.5: accuracy of CV
= 0.19%, accuracy of Train = 0.16%,
 learning rate = 0.6500000000000001: Regularisation Strength = 0.6: accuracy of CV
= 0.19%, accuracy of Train = 0.17%,
 learning rate = 0.6500000000000001: Regularisation Strength = 0.7: accuracy of CV
= 0.17%, accuracy of Train = 0.17%,
 learning rate = 0.6500000000000001: Regularisation Strength = 0.8: accuracy of CV
= 0.19%, accuracy of Train = 0.20%,
 learning rate = 0.6500000000000001: Regularisation Strength = 0.9: accuracy of CV
= 0.12%, accuracy of Train = 0.14%,
 learning rate = 0.6500000000000001: Regularisation Strength = 1: accuracy of CV =
0.13%, accuracy of Train = 0.14%,
 learning rate = 0.7000000000000001: Regularisation Strength = 0.2: accuracy of CV
= 0.21%, accuracy of Train = 0.17%,
 learning rate = 0.7000000000000001: Regularisation Strength = 0.3: accuracy of CV
= 0.18%, accuracy of Train = 0.14%,
 learning rate = 0.7000000000000001: Regularisation Strength = 0.4: accuracy of CV
= 0.18%, accuracy of Train = 0.15%,
 learning rate = 0.7000000000000001: Regularisation Strength = 0.5: accuracy of CV
= 0.19%, accuracy of Train = 0.17%,
 learning rate = 0.7000000000000001: Regularisation Strength = 0.6: accuracy of CV
= 0.19%, accuracy of Train = 0.17%,
 learning rate = 0.7000000000000001: Regularisation Strength = 0.7: accuracy of CV
= 0.17%, accuracy of Train = 0.17%,
 learning rate = 0.7000000000000001: Regularisation Strength = 0.8: accuracy of CV
= 0.18%, accuracy of Train = 0.20%,
 learning rate = 0.7000000000000001: Regularisation Strength = 0.9: accuracy of CV
= 0.13%, accuracy of Train = 0.14%,
 learning rate = 0.7000000000000001: Regularisation Strength = 1: accuracy of CV =

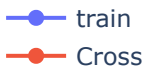
0.13%, accuracy of Train = 0.14%,
 learning rate = 0.7500000000000001: Regularisation Strength = 0.2: accuracy of CV
= 0.17%, accuracy of Train = 0.14%,
 learning rate = 0.7500000000000001: Regularisation Strength = 0.3: accuracy of CV
= 0.17%, accuracy of Train = 0.14%,
 learning rate = 0.7500000000000001: Regularisation Strength = 0.4: accuracy of CV
= 0.18%, accuracy of Train = 0.15%,
 learning rate = 0.7500000000000001: Regularisation Strength = 0.5: accuracy of CV
= 0.19%, accuracy of Train = 0.18%,
 learning rate = 0.7500000000000001: Regularisation Strength = 0.6: accuracy of CV
= 0.18%, accuracy of Train = 0.17%,
 learning rate = 0.7500000000000001: Regularisation Strength = 0.7: accuracy of CV
= 0.19%, accuracy of Train = 0.20%,
 learning rate = 0.7500000000000001: Regularisation Strength = 0.8: accuracy of CV
= 0.13%, accuracy of Train = 0.14%,
 learning rate = 0.7500000000000001: Regularisation Strength = 0.9: accuracy of CV
= 0.13%, accuracy of Train = 0.14%,
 learning rate = 0.7500000000000001: Regularisation Strength = 1: accuracy of CV =
0.13%, accuracy of Train = 0.14%,
 learning rate = 0.8: Regularisation Strength = 0.2: accuracy of CV = 0.17%, accura
cy of Train = 0.14%,
 learning rate = 0.8: Regularisation Strength = 0.3: accuracy of CV = 0.18%, accura
cy of Train = 0.14%,
 learning rate = 0.8: Regularisation Strength = 0.4: accuracy of CV = 0.19%, accura
cy of Train = 0.16%,
 learning rate = 0.8: Regularisation Strength = 0.5: accuracy of CV = 0.19%, accura
cy of Train = 0.17%,
 learning rate = 0.8: Regularisation Strength = 0.6: accuracy of CV = 0.17%, accura
cy of Train = 0.17%,
 learning rate = 0.8: Regularisation Strength = 0.7: accuracy of CV = 0.13%, accura
cy of Train = 0.13%,
 learning rate = 0.8: Regularisation Strength = 0.8: accuracy of CV = 0.13%, accura
cy of Train = 0.14%,
 learning rate = 0.8: Regularisation Strength = 0.9: accuracy of CV = 0.13%, accura
cy of Train = 0.14%,
 learning rate = 0.8: Regularisation Strength = 1: accuracy of CV = 0.13%, accuracy
of Train = 0.14%,
 learning rate = 0.8500000000000001: Regularisation Strength = 0.2: accuracy of CV
= 0.17%, accuracy of Train = 0.14%,
 learning rate = 0.8500000000000001: Regularisation Strength = 0.3: accuracy of CV
= 0.17%, accuracy of Train = 0.15%,
 learning rate = 0.8500000000000001: Regularisation Strength = 0.4: accuracy of CV
= 0.19%, accuracy of Train = 0.17%,
 learning rate = 0.8500000000000001: Regularisation Strength = 0.5: accuracy of CV
= 0.18%, accuracy of Train = 0.17%,
 learning rate = 0.8500000000000001: Regularisation Strength = 0.6: accuracy of CV
= 0.19%, accuracy of Train = 0.20%,
 learning rate = 0.8500000000000001: Regularisation Strength = 0.7: accuracy of CV
= 0.13%, accuracy of Train = 0.14%,
 learning rate = 0.8500000000000001: Regularisation Strength = 0.8: accuracy of CV
= 0.14%, accuracy of Train = 0.14%,
 learning rate = 0.8500000000000001: Regularisation Strength = 0.9: accuracy of CV
= 0.13%, accuracy of Train = 0.14%,
 learning rate = 0.8500000000000001: Regularisation Strength = 1: accuracy of CV =
0.12%, accuracy of Train = 0.13%,
 learning rate = 0.9000000000000001: Regularisation Strength = 0.2: accuracy of CV

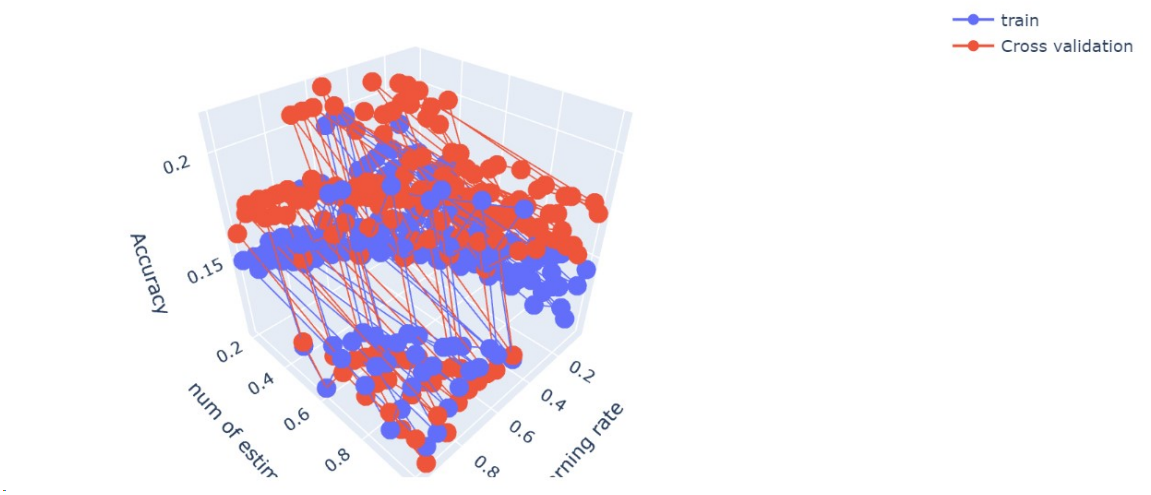
```
= 0.17%, accuracy of Train = 0.15%,
 learning rate = 0.9000000000000001:  Regularisation Strength = 0.3: accuracy of CV
= 0.17%, accuracy of Train = 0.15%,
 learning rate = 0.9000000000000001:  Regularisation Strength = 0.4: accuracy of CV
= 0.19%, accuracy of Train = 0.17%,
 learning rate = 0.9000000000000001:  Regularisation Strength = 0.5: accuracy of CV
= 0.17%, accuracy of Train = 0.17%,
 learning rate = 0.9000000000000001:  Regularisation Strength = 0.6: accuracy of CV
= 0.18%, accuracy of Train = 0.20%,
 learning rate = 0.9000000000000001:  Regularisation Strength = 0.7: accuracy of CV
= 0.13%, accuracy of Train = 0.14%,
 learning rate = 0.9000000000000001:  Regularisation Strength = 0.8: accuracy of CV
= 0.13%, accuracy of Train = 0.14%,
 learning rate = 0.9000000000000001:  Regularisation Strength = 0.9: accuracy of CV
= 0.13%, accuracy of Train = 0.13%,
 learning rate = 0.9000000000000001:  Regularisation Strength = 1: accuracy of CV =
0.13%, accuracy of Train = 0.12%,
 learning rate = 0.9500000000000001:  Regularisation Strength = 0.2: accuracy of CV
= 0.17%, accuracy of Train = 0.15%,
 learning rate = 0.9500000000000001:  Regularisation Strength = 0.3: accuracy of CV
= 0.18%, accuracy of Train = 0.15%,
 learning rate = 0.9500000000000001:  Regularisation Strength = 0.4: accuracy of CV
= 0.19%, accuracy of Train = 0.17%,
 learning rate = 0.9500000000000001:  Regularisation Strength = 0.5: accuracy of CV
= 0.17%, accuracy of Train = 0.17%,
 learning rate = 0.9500000000000001:  Regularisation Strength = 0.6: accuracy of CV
= 0.13%, accuracy of Train = 0.13%,
 learning rate = 0.9500000000000001:  Regularisation Strength = 0.7: accuracy of CV
= 0.13%, accuracy of Train = 0.14%,
 learning rate = 0.9500000000000001:  Regularisation Strength = 0.8: accuracy of CV
= 0.13%, accuracy of Train = 0.14%,
 learning rate = 0.9500000000000001:  Regularisation Strength = 0.9: accuracy of CV
= 0.12%, accuracy of Train = 0.13%,
 learning rate = 0.9500000000000001:  Regularisation Strength = 1: accuracy of CV =
0.11%, accuracy of Train = 0.12%,
 learning rate = 1.0:  Regularisation Strength = 0.2: accuracy of CV = 0.16%, accura
cy of Train = 0.15%,
 learning rate = 1.0:  Regularisation Strength = 0.3: accuracy of CV = 0.18%, accura
cy of Train = 0.15%,
 learning rate = 1.0:  Regularisation Strength = 0.4: accuracy of CV = 0.18%, accura
cy of Train = 0.17%,
 learning rate = 1.0:  Regularisation Strength = 0.5: accuracy of CV = 0.13%, accura
cy of Train = 0.13%,
 learning rate = 1.0:  Regularisation Strength = 0.6: accuracy of CV = 0.11%, accura
cy of Train = 0.11%,
 learning rate = 1.0:  Regularisation Strength = 0.7: accuracy of CV = 0.13%, accura
cy of Train = 0.14%,
 learning rate = 1.0:  Regularisation Strength = 0.8: accuracy of CV = 0.13%, accura
cy of Train = 0.13%,
 learning rate = 1.0:  Regularisation Strength = 0.9: accuracy of CV = 0.13%, accura
cy of Train = 0.12%,
 learning rate = 1.0:  Regularisation Strength = 1: accuracy of CV = 0.13%, accuracy
of Train = 0.17%,
```

In [164…

```
trace1 = go.Scatter3d(x=learning_rate,y=regularization_strength,z= train_accuracy_s
trace2 = go.Scatter3d(x=learning_rate,y=regularization_strength,z=val_accuracy_scor
```

```
data = [trace1, trace2]

layout = go.Layout(scene = dict(
        xaxis = dict(title='learning rate'),
        yaxis = dict(title='num of estimators'),
        zaxis = dict(title='Accuracy'),))

fig = go.Figure(data=data, layout=layout)
offline.iplot(fig, filename='3d-scatter-colorscale')
```



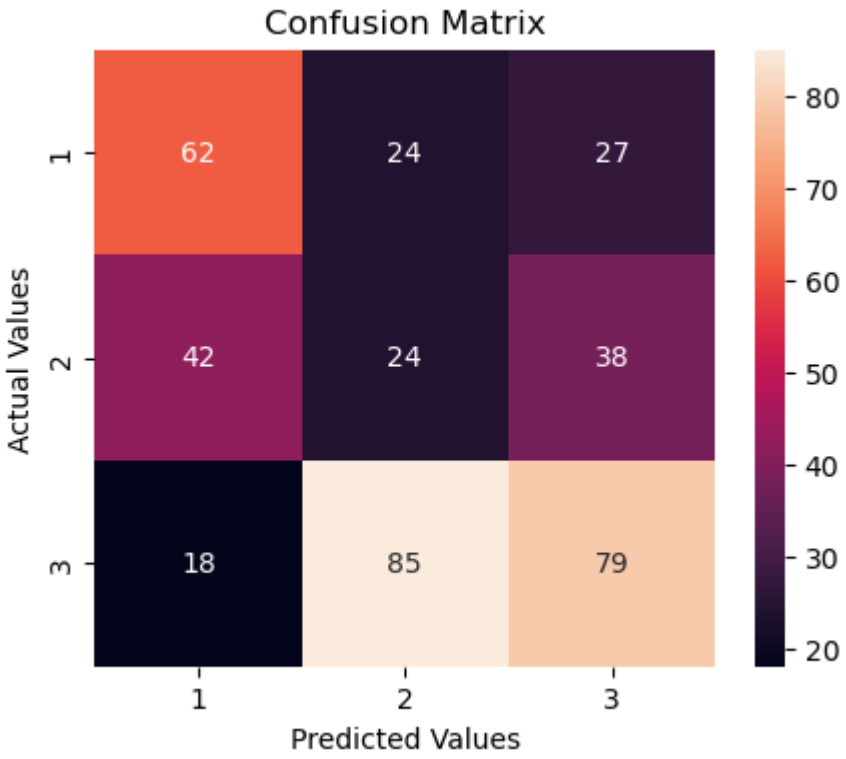


In [165…
```
print('Best regularisation strength is',0.2,'Best learning rate ',0.65)
#print(accuracy_score(bdt_real.predict(X_test),y_test))
svm_test = MulticlassSVM(learning_rate = 0.65,regularization_strength =0.2 )
svm_test.fit(X_train,y_train)
test_pred = svm_test.predict(X_test)
print(accuracy_score(test_pred,y_test))
```

```
Best regularisation strength is 0.2 Best learning rate  0.65
0.15538847117794485
```

In [166…
```
final_pred = []
for i in test_pred:
        final_pred.append(i+1)
```

In [167…
```
cm = confusion_matrix(y_test, final_pred)
cm_df = pd.DataFrame(cm,index = ['1','2','3'], columns = ['1','2','3'])
plt.figure(figsize=(5,4))
sns.heatmap(cm_df, annot=True)
plt.title('Confusion Matrix')
plt.ylabel('Actual Values')
plt.xlabel('Predicted Values')
plt.show()
```



In [168…

```
tpforclass1 = cm[0][0]
fnforclass1 = cm[0][1] + cm[0][2]
tnforclass1 = cm[1][1] + cm[1][2] + cm[2][1] + cm[2][2]
fpforclass1 = cm[1][0] + cm[2][0]

recallforclass1 = (tpforclass1) / (tpforclass1 + fnforclass1)
precisionforclass1 = (tpforclass1) / (tpforclass1 + fpforclass1)
f1scoreforclass1 = (2 * precisionforclass1 * recallforclass1) / (precisionforclass1

tpforclass2 = cm[1][1]
fnforclass2 = cm[1][0] + cm[1][2]
tnforclass2 = cm[0][0] + cm[0][2] + cm[2][0] + cm[2][2]
fpforclass2 = cm[0][1] + cm[2][1]

recallforclass2 = (tpforclass2) / (tpforclass2 + fnforclass2)
precisionforclass2 = (tpforclass2) / (tpforclass2 + fpforclass2)
f1scoreforclass2 = (2 * precisionforclass2 * recallforclass2) / (precisionforclass2

tpforclass3 = cm[2][2]
fnforclass3 = cm[1][0] + cm[1][2]
tnforclass3 = cm[0][0] + cm[0][2] + cm[2][0] + cm[2][2]
fpforclass3 = cm[0][1] + cm[2][1]

recallforclass3 = (tpforclass3) / (tpforclass3 + fnforclass3)
precisionforclass3 = (tpforclass3) / (tpforclass3 + fpforclass3)
f1scoreforclass3 = (2 * precisionforclass3 * recallforclass3) / (precisionforclass3

print("Values of Class 1 evaluation metrics ---------- \n")
print("Precision of Class 1 is ")
print(precisionforclass1)
print("Recall of Class 1 is ")
print(recallforclass1)
```

```
print("F1-Score of Class 1 is ")
print(f1scoreforclass1)
print("Values of Class 2 evaluation metrics ---------- \n")
print("Precision of Class 2 is ")
print(precisionforclass2)
print("Recall of Class 2 is ")
print(recallforclass2)
print("F1-Score of Class 2 is ")
print(f1scoreforclass2)
print("Values of Class 3 evaluation metrics ---------- \n")
print("Precision of Class 3 is ")
print(precisionforclass3)
print("Recall of Class 3 is ")
print(recallforclass3)
print("F1-Score of Class 3 is ")
print(f1scoreforclass3)
```

```
Values of Class 1 evaluation metrics ----------

Precision of Class 1 is
0.5081967213114754
Recall of Class 1 is
0.5486725663716814
F1-Score of Class 1 is
0.5276595744680852
Values of Class 2 evaluation metrics ----------

Precision of Class 2 is
0.18045112781954886
Recall of Class 2 is
0.23076923076923078
F1-Score of Class 2 is
0.20253164556962025
Values of Class 3 evaluation metrics ----------

Precision of Class 3 is
0.42021276595744683
Recall of Class 3 is
0.4968553459119497
F1-Score of Class 3 is
0.45533141210374645
```

4.2 Insights drawn (plots, markdown explanations)

1) We have used validation data in order to find the correct value of the hyperparameters Regularization parameter and Learning rate.
2) From the train accuracy and cross validation accuracy plots we can see that as the Regularization parameter and Learning rate value increases, the train accuracy gets higher while cross-validation accuracy remains almost same indicating the occurence of overfitting with the increase in hyperparameter values.
3) At Regularization parameter = 0.2 and learning rate = 0.65, there is a proper balance between Train accuracy and Cross-Validation accuracy which proves that it is the best value

using which the performance of the model can be evaluated on unseen data.
3) The confusion matrix has less high values along the major diagonal indicating that the model is able to classify a certain number of unseen datapoints accurately.
4) The F1-Score for all the classes are < 0.5 indicating that the performance of the model on unseen dataset is not appreciable.

Comparison of performance between 3 Models

By comparing the confusion matrix and F1-Scores of the 3 models we can conclude that the performance of the Decision Tree on the unseen dataset is the best, followed by Adaboost and then Multiclass SVM. Multiclass SVM has the worst performance when compared with the three models. This can be attributed to the fact that normal one vs rest SVM classifiers are linear classification models. There may be some datapoints which may be present in the regions of ambiguity which cannot be correctly classified by SVM resulting in the degradation in its performance.

5. References

1. Journal Article "Multi-class adaboost" Hastie, Trevor,A Rosset, Saharonhu, Ji Zou, Hui. Internation Journal of Statistics and its Interface Volume 2 Pages 349-360 1938-7997 International Press of Boston.
2. https://hastie.su.domains/Papers/samme.pdf
3. Multiclass Classification Using SVM : https://www.analyticsvidhya.com/blog/2021/05/multiclass-classification-using-svm/
4. Multiclass Classification with Support Vector Machines (SVM), Dual Problem and Kernel Functions: https://towardsdatascience.com/multiclass-classification-with-support-vector-machines-svm-kernel-trick-kernel-functions-f9d5377d6f02
5. Decision Tree Algorithm for Multiclass problems: https://towardsdatascience.com/decision-tree-algorithm-for-multiclass-problems-using-python-6b0ec1183bf5
6. Multi-class Classification using Decision-Tree Model: https://adityagoel123.medium.com/multi-class-classification-using-decision-tree-model-68e75114303

In [ ]:

In [ ]: