**Spring 2023: CS5710 – Machine Learning**

In-Class Programming Assignment-4

GitHub Link - <https://github.com/raimukul/MachineLearning_Assignments>

Video link- <https://drive.google.com/file/d/12RFhEuUN1nxvux8EN39QYeSSReYUybeZ/view?usp=sharing>

Code:

**1. Pandas**

1. Read the provided CSV file ‘data.csv’.

<https://drive.google.com/drive/folders/1h8C3mLsso-R-sIOLsvoYwPLzy2fJ4IOF?usp=sharing>

2. Show the basic statistical description of the data.

3. Check if the data has null values. a. Replace the null values with the mean.

4. Select at least two columns and aggregate the data using: min, max, count, mean.

5. Filter the data frame to select the rows with calories values between 500 and 1000.

6. Filter the data frame to select the rows with calories values > 500 and pulse < 100.

7. Create a new “df\_modified” data frame that contains all the columns from df except for “Maxpulse”.

8. Delete the “Maxpulse” column from the main df dataframe

9. Convert the datatype of Calories column to int datatype.

10. Using pandas create a scatter plot for the two columns (Duration and Calories).

In [1]:

*#Read the provided CSV file ‘data.csv’. https://drive.google.com/drive/folders/1h8C3mLsso-R-sIOLsvoYwPLzy2fJ4IOF?usp=sharing*

import pandas as pd

df = pd.read\_csv('/content/drive/MyDrive/Colab Notebooks/data/data.csv')

In [17]:

print(df)

Duration Pulse Maxpulse Calories

0 60 110 130 409.1

1 60 117 145 479.0

2 60 103 135 340.0

3 45 109 175 282.4

4 45 117 148 406.0

.. ... ... ... ...

164 60 105 140 290.8

165 60 110 145 300.0

166 60 115 145 310.2

167 75 120 150 320.4

168 75 125 150 330.4

[169 rows x 4 columns]

In [2]:

df = pd.DataFrame(df)

In [3]:

*#Show the basic statistical description about the data.*

df=df.describe()

df

Out[3]:

|  | **Duration** | **Pulse** | **Maxpulse** | **Calories** |
| --- | --- | --- | --- | --- |
| **count** | 169.000000 | 169.000000 | 169.000000 | 164.000000 |
| **mean** | 63.846154 | 107.461538 | 134.047337 | 375.790244 |
| **std** | 42.299949 | 14.510259 | 16.450434 | 266.379919 |
| **min** | 15.000000 | 80.000000 | 100.000000 | 50.300000 |
| **25%** | 45.000000 | 100.000000 | 124.000000 | 250.925000 |
| **50%** | 60.000000 | 105.000000 | 131.000000 | 318.600000 |
| **75%** | 60.000000 | 111.000000 | 141.000000 | 387.600000 |
| **max** | 300.000000 | 159.000000 | 184.000000 | 1860.400000 |

In [4]:

*#Check if the data has null values.*

df = pd.read\_csv('/content/drive/MyDrive/Colab Notebooks/data/data.csv')

df.isnull()

Out[4]:

|  | **Duration** | **Pulse** | **Maxpulse** | **Calories** |
| --- | --- | --- | --- | --- |
| **0** | False | False | False | False |
| **1** | False | False | False | False |
| **2** | False | False | False | False |
| **3** | False | False | False | False |
| **4** | False | False | False | False |
| **...** | ... | ... | ... | ... |
| **164** | False | False | False | False |
| **165** | False | False | False | False |
| **166** | False | False | False | False |
| **167** | False | False | False | False |
| **168** | False | False | False | False |

169 rows × 4 columns

In [5]:

*#checking is there any null value is there or not.*

df.isnull().values.any()

Out[5]:

True

In [7]:

*# a. Replace the null values with the mean*

new\_df=df.fillna(df.mean())

In [10]:

new\_df.isnull().values.any()

Out[10]:

False

In [11]:

*#4. Select at least two columns and aggregate the data using: min, max, count, mean*

*# by using groupby function with aggregation to get mean, min and max values*

result = df.groupby('Duration').agg({'Calories': ['mean', 'min', 'max']})

print("Mean, min, and max values are")

print(result)

Mean, min, and max values are

Calories

mean min max

Duration

15 87.350000 50.5 124.2

20 151.600000 50.3 229.4

25 244.200000 244.2 244.2

30 192.125000 86.2 319.2

45 273.236364 100.7 406.0

60 339.675000 215.2 486.0

75 325.400000 320.4 330.4

80 643.100000 643.1 643.1

90 541.800000 466.4 700.0

120 666.833333 500.0 1000.1

150 939.400000 816.0 1115.0

160 943.700000 853.0 1034.4

180 733.600000 600.1 800.4

210 1618.200000 1376.0 1860.4

270 1729.000000 1729.0 1729.0

300 1500.200000 1500.2 1500.2

In [12]:

*#5. Filter the dataframe to select the rows with calories values between 500 and 1000.*

df.query('Calories < 1000 and Calories > 500')

Out[12]:

|  | **Duration** | **Pulse** | **Maxpulse** | **Calories** |
| --- | --- | --- | --- | --- |
| **51** | 80 | 123 | 146 | 643.1 |
| **62** | 160 | 109 | 135 | 853.0 |
| **65** | 180 | 90 | 130 | 800.4 |
| **66** | 150 | 105 | 135 | 873.4 |
| **67** | 150 | 107 | 130 | 816.0 |
| **72** | 90 | 100 | 127 | 700.0 |
| **73** | 150 | 97 | 127 | 953.2 |
| **75** | 90 | 98 | 125 | 563.2 |
| **78** | 120 | 100 | 130 | 500.4 |
| **90** | 180 | 101 | 127 | 600.1 |
| **99** | 90 | 93 | 124 | 604.1 |
| **103** | 90 | 90 | 100 | 500.4 |
| **106** | 180 | 90 | 120 | 800.3 |
| **108** | 90 | 90 | 120 | 500.3 |

In [13]:

*# 6. Filter the dataframe to select the rows with calories values > 500 and pulse < 100*

df.query('Calories > 500 and Pulse < 100')

Out[13]:

|  | **Duration** | **Pulse** | **Maxpulse** | **Calories** |
| --- | --- | --- | --- | --- |
| **65** | 180 | 90 | 130 | 800.4 |
| **70** | 150 | 97 | 129 | 1115.0 |
| **73** | 150 | 97 | 127 | 953.2 |
| **75** | 90 | 98 | 125 | 563.2 |
| **99** | 90 | 93 | 124 | 604.1 |
| **103** | 90 | 90 | 100 | 500.4 |
| **106** | 180 | 90 | 120 | 800.3 |
| **108** | 90 | 90 | 120 | 500.3 |

In [14]:

*#7. Create a new “df\_modified” dataframe that contains all the columns from df except for “Maxpulse”*

df\_modified=df.drop(columns=["Maxpulse"])

df\_modified

Out[14]:

|  | **Duration** | **Pulse** | **Calories** |
| --- | --- | --- | --- |
| **0** | 60 | 110 | 409.1 |
| **1** | 60 | 117 | 479.0 |
| **2** | 60 | 103 | 340.0 |
| **3** | 45 | 109 | 282.4 |
| **4** | 45 | 117 | 406.0 |
| **...** | ... | ... | ... |
| **164** | 60 | 105 | 290.8 |
| **165** | 60 | 110 | 300.0 |
| **166** | 60 | 115 | 310.2 |
| **167** | 75 | 120 | 320.4 |
| **168** | 75 | 125 | 330.4 |

169 rows × 3 columns

In [17]:

*# 8. Delete the “Maxpulse” column from the main df dataframe*

df.drop(columns=["Maxpulse"], axis=1, inplace=True)

df

Out[17]:

|  | **Duration** | **Pulse** | **Calories** |
| --- | --- | --- | --- |
| **0** | 60 | 110 | 409.1 |
| **1** | 60 | 117 | 479.0 |
| **2** | 60 | 103 | 340.0 |
| **3** | 45 | 109 | 282.4 |
| **4** | 45 | 117 | 406.0 |
| **...** | ... | ... | ... |
| **164** | 60 | 105 | 290.8 |
| **165** | 60 | 110 | 300.0 |
| **166** | 60 | 115 | 310.2 |
| **167** | 75 | 120 | 320.4 |
| **168** | 75 | 125 | 330.4 |

169 rows × 3 columns

In [22]:

*#9. Convert the datatype of Calories column to int datatype.*

df=df.fillna(df.mean())

df = df.astype({'Calories':'int'})

print(df.dtypes)

Duration int64

Pulse int64

Calories int64

dtype: object

In [23]:

*#Using pandas create a scatter plot for the two columns (Duration and Calories).*

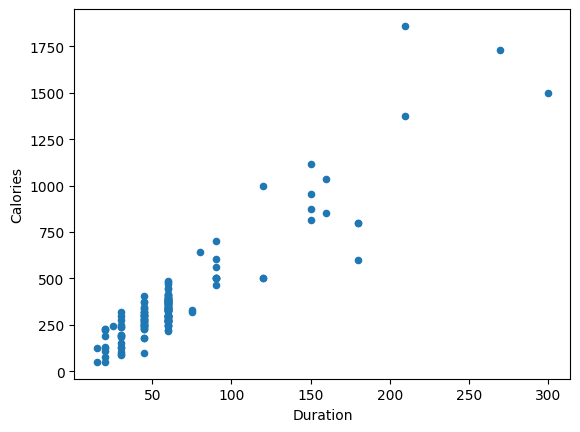
df.plot(kind = 'scatter', x = 'Duration', y = 'Calories')

/usr/local/lib/python3.9/dist-packages/pandas/plotting/\_matplotlib/core.py:1114: UserWarning: No data for colormapping provided via 'c'. Parameters 'cmap' will be ignored

scatter = ax.scatter(

Out[23]:

<Axes: xlabel='Duration', ylabel='Calories'>



## 1. (Titanic Dataset)

1. Find the correlation between ‘survived’ (target column) and ‘sex’ column for the Titanic use case inclass.

a. Do you think we should keep this feature?

2. Do at least two visualizations to describe or show correlations.

3. Implement Naïve Bayes method using scikit-learn library and report the accuracy.

In [24]:

*#import data*

test\_df = pd.read\_csv("/content/drive/MyDrive/Colab Notebooks/Dataset/test.csv")

train\_df = pd.read\_csv("/content/drive/MyDrive/Colab Notebooks/Dataset/train.csv")

In [25]:

*#Data ANalysis*

train\_df.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 891 entries, 0 to 890

Data columns (total 12 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 PassengerId 891 non-null int64

1 Survived 891 non-null int64

2 Pclass 891 non-null int64

3 Name 891 non-null object

4 Sex 891 non-null object

5 Age 714 non-null float64

6 SibSp 891 non-null int64

7 Parch 891 non-null int64

8 Ticket 891 non-null object

9 Fare 891 non-null float64

10 Cabin 204 non-null object

11 Embarked 889 non-null object

dtypes: float64(2), int64(5), object(5)

memory usage: 83.7+ KB

In [26]:

train\_df.describe()

Out[26]:

|  | **PassengerId** | **Survived** | **Pclass** | **Age** | **SibSp** | **Parch** | **Fare** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **count** | 891.000000 | 891.000000 | 891.000000 | 714.000000 | 891.000000 | 891.000000 | 891.000000 |
| **mean** | 446.000000 | 0.383838 | 2.308642 | 29.699118 | 0.523008 | 0.381594 | 32.204208 |
| **std** | 257.353842 | 0.486592 | 0.836071 | 14.526497 | 1.102743 | 0.806057 | 49.693429 |
| **min** | 1.000000 | 0.000000 | 1.000000 | 0.420000 | 0.000000 | 0.000000 | 0.000000 |
| **25%** | 223.500000 | 0.000000 | 2.000000 | 20.125000 | 0.000000 | 0.000000 | 7.910400 |
| **50%** | 446.000000 | 0.000000 | 3.000000 | 28.000000 | 0.000000 | 0.000000 | 14.454200 |
| **75%** | 668.500000 | 1.000000 | 3.000000 | 38.000000 | 1.000000 | 0.000000 | 31.000000 |
| **max** | 891.000000 | 1.000000 | 3.000000 | 80.000000 | 8.000000 | 6.000000 | 512.329200 |

In [27]:

import seaborn as sns

%matplotlib inline

from matplotlib import pyplot as plt

survived = 'Survived'

not\_survived = 'not survived'

fig, axes = plt.subplots(nrows=1, ncols=2,figsize=(10, 4))

women = train\_df[train\_df['Sex']=='female']

men = train\_df[train\_df['Sex']=='male']

ax = sns.distplot(women[women['Survived']==1].Age.dropna(), bins=18, label = survived, ax = axes[0], kde =False)

ax = sns.distplot(women[women['Survived']==0].Age.dropna(), bins=40, label = not\_survived, ax = axes[0], kde =False)

ax.legend()

ax.set\_title('Female')

ax = sns.distplot(men[men['Survived']==1].Age.dropna(), bins=18, label = survived, ax = axes[1], kde = False)

ax = sns.distplot(men[men['Survived']==0].Age.dropna(), bins=40, label = not\_survived, ax = axes[1], kde = False)

ax.legend()

\_ = ax.set\_title('Male')

<ipython-input-27-0cf8acbfe0d6>:10: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with

similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see

https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

ax = sns.distplot(women[women['Survived']==1].Age.dropna(), bins=18, label = survived, ax = axes[0], kde =False)

<ipython-input-27-0cf8acbfe0d6>:11: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with

similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see

https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

ax = sns.distplot(women[women['Survived']==0].Age.dropna(), bins=40, label = not\_survived, ax = axes[0], kde =False)

<ipython-input-27-0cf8acbfe0d6>:14: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with

similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see

https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

ax = sns.distplot(men[men['Survived']==1].Age.dropna(), bins=18, label = survived, ax = axes[1], kde = False)

<ipython-input-27-0cf8acbfe0d6>:15: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

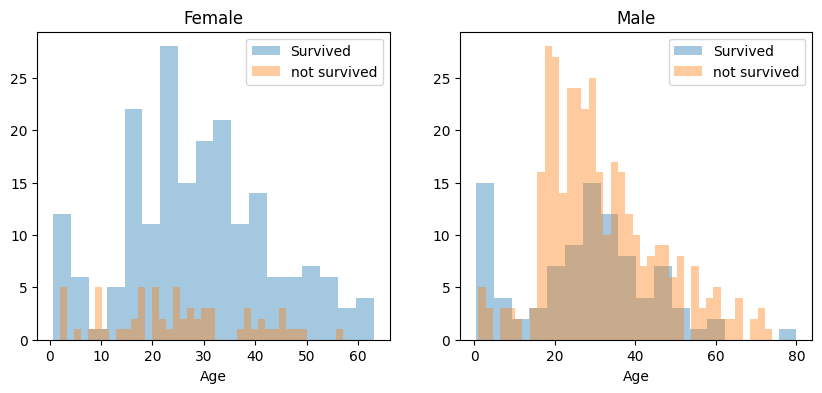
Please adapt your code to use either `displot` (a figure-level function with

similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see

https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

ax = sns.distplot(men[men['Survived']==0].Age.dropna(), bins=40, label = not\_survived, ax = axes[1], kde = False)



In [30]:

from matplotlib import pyplot as plt

import pandas as pd

import numpy as np

from sklearn.naive\_bayes import GaussianNB

from sklearn.metrics import accuracy\_score

train\_data = pd.read\_csv('/content/drive/MyDrive/Colab Notebooks/Dataset/train.csv')

test\_data = pd.read\_csv('/content/drive/MyDrive/Colab Notebooks/Dataset/test.csv')

*# Find the correlation between Survived and Sex*

corr = train\_data['Survived'].corr(train\_data['Sex'].astype('category').cat.codes)

print("Correlation between Survived and Sex: ",corr)

print('a. Do you think we should keep this feature?')

print('Yes, we should keep this feature as it has a correlation of', corr, 'with the target variable but some other features can be dropped as they have very less correlation with the target variable.')

*# Do at least two visualizations to describe the data*

*# Histogram of age*

train\_data['Age'].plot.hist(title='Histogram of Age')

plt.show()

*# Scatter plot of age and fare*

train\_data.plot.scatter(x='Age', y='Fare', title='Scatter plot of Age and Fare')

plt.show()

*# Plot between age and survived*

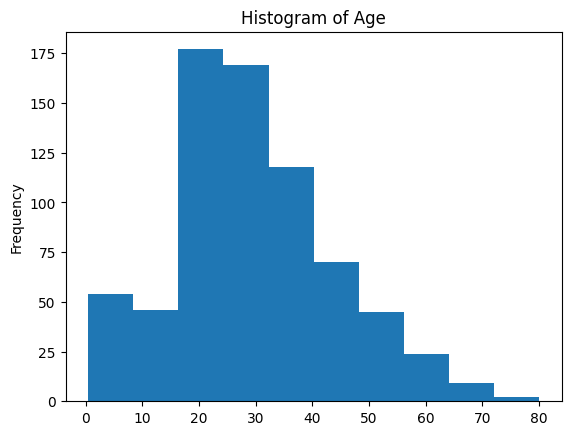
train\_data.plot.scatter(x='Age', y='Survived', title='Scatter plot of Age and Survived')

plt.show()

Correlation between Survived and Sex: -0.5433513806577555

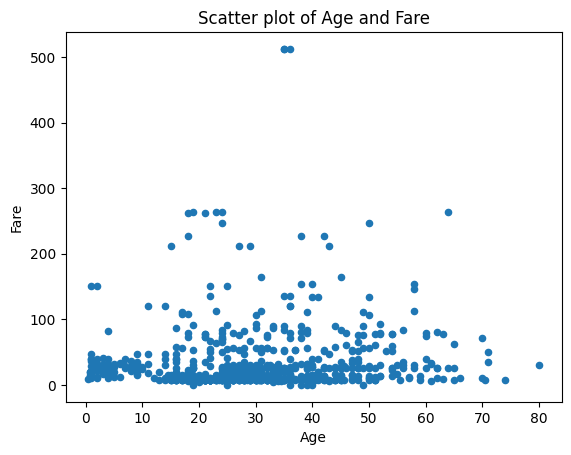
a. Do you think we should keep this feature?

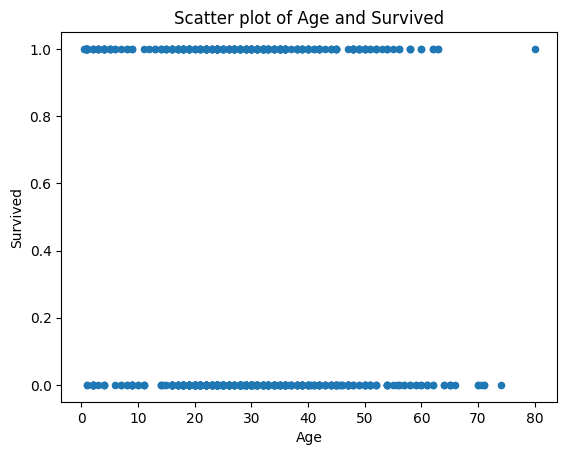
Yes, we should keep this feature as it has a correlation of -0.5433513806577555 with the target variable but some other features can be dropped as they have very less correlation with the target variable.



/usr/local/lib/python3.9/dist-packages/pandas/plotting/\_matplotlib/core.py:1114: UserWarning: No data for colormapping provided via 'c'. Parameters 'cmap' will be ignored

scatter = ax.scatter(





## 2. (Glass Dataset)

1. Implement Naïve Bayes method using scikit-learn library. a. Use the glass dataset available in Link also provided in your assignment. b. Use train\_test\_split to create training and testing part.
2. Evaluate the model on testing part using score and Do at least two visualizations to describe or show correlations in the Glass Dataset.

In [32]:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.naive\_bayes import GaussianNB

from sklearn.svm import SVC

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import classification\_report, accuracy\_score

import warnings

warnings.filterwarnings("ignore")

*# a. read glass.csv file as a dataframe*

glass\_data = pd.read\_csv('/content/drive/MyDrive/Colab Notebooks/Dataset/glass.csv')

*# b. Use train\_test\_split to create training and testing part*

*# Split the data into training and testing data*

X\_train, X\_test, y\_train, y\_test = train\_test\_split(glass\_data.drop('Type', axis=1), glass\_data['Type'], test\_size=0.3, random\_state=42)

*# implement Naïve Bayes method using scikit-learn library and Evaluate the model on testing part using score and classification\_report*

*# Create a Gaussian Classifier*

model = GaussianNB()

*# Train the model using the training sets*

model.fit(X\_train, y\_train)

*# Predict the response for test dataset*

y\_pred = model.predict(X\_test)

*# Calculate the accuracy of the model*

print("Accuracy of the Naive Bayes model: ", accuracy\_score(y\_test, y\_pred))

*# Print classification report*

print('Classification Report for Naive Bayes model: ')

print(classification\_report(y\_test, y\_pred))

*# Use SVM method using scikit-learn library and Evaluate the model on testing part using score and classification\_report*

*# Create a SVM Classifier*

model = SVC(kernel='linear')

*# Train the model using the training sets*

model.fit(X\_train, y\_train)

*# Predict the response for test dataset*

y\_pred = model.predict(X\_test)

*# Calculate the accuracy of the model*

print("Accuracy of the SVM model: ", accuracy\_score(y\_test, y\_pred))

*# Print classification report*

print('Classification Report for SVM model: ')

print(classification\_report(y\_test, y\_pred))

*# Do at least two visualizations to describe or show correlations in the Glass Dataset*

*# Histogram of refractive index*

glass\_data['RI'].plot.hist(title='Histogram of Refractive Index')

plt.show()

*# Scatter plot of refractive index and Ca*

glass\_data.plot.scatter(x='RI', y='Ca', title='Scatter plot of Refractive Index and Ca')

plt.show()

print('Which algorithm you got better accuracy? Can you justify why?')

*# accuracy of Naive Bayes model: 0.3076923076923077*

*# accuracy of SVM model: 0.676923076923077*

print('SVM model has better accuracy than Naive Bayes model. This is because SVM model tries to find the best possible decision boundary between the data points of different classes. It tries to maximize the margin between the decision boundary and the data points. On the other hand, Naive Bayes model assumes that the features are independent of each other and tries to find the probability of the data point belonging to a particular class. Hence, SVM model has better accuracy than Naive Bayes model.')

Accuracy of the Naive Bayes model: 0.3076923076923077

Classification Report for Naive Bayes model:

precision recall f1-score support

1 0.00 0.00 0.00 19

2 0.40 0.17 0.24 23

3 0.08 0.75 0.15 4

5 0.33 0.17 0.22 6

6 0.75 1.00 0.86 3

7 0.90 0.90 0.90 10

accuracy 0.31 65

macro avg 0.41 0.50 0.40 65

weighted avg 0.35 0.31 0.29 65

Accuracy of the SVM model: 0.676923076923077

Classification Report for SVM model:

precision recall f1-score support

1 0.65 0.79 0.71 19

2 0.59 0.70 0.64 23

3 0.00 0.00 0.00 4

5 0.75 0.50 0.60 6

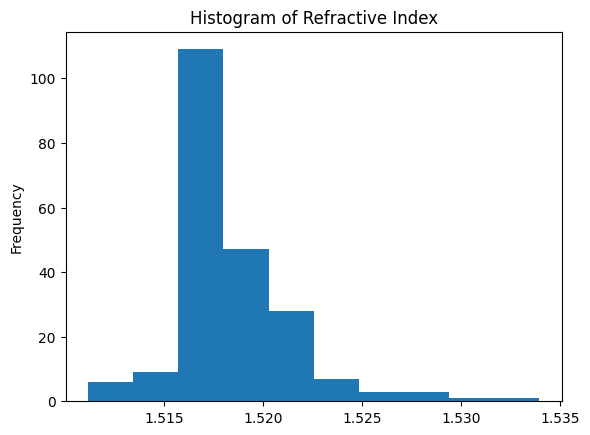
6 0.50 0.33 0.40 3

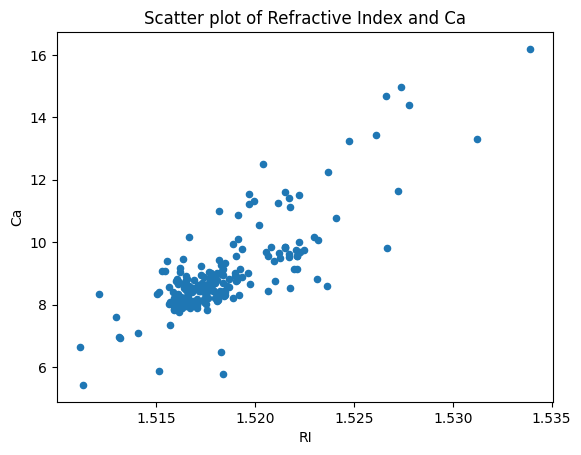
7 1.00 0.90 0.95 10

accuracy 0.68 65

macro avg 0.58 0.54 0.55 65

weighted avg 0.65 0.68 0.65 65





Which algorithm you got better accuracy? Can you justify why?

SVM model has better accuracy than Naive Bayes model. This is because SVM model tries to find the best possible decision boundary between the data points of different classes. It tries to maximize the margin between the decision boundary and the data points. On the other hand, Naive Bayes model assumes that the features are independent of each other and tries to find the probability of the data point belonging to a particular class. Hence, SVM model has better accuracy than Naive Bayes model.