In [1]:
```
import pandas as pd
import numpy as np
import warnings
from operator import add
warnings.filterwarnings('ignore')
```

Loading Dataset

In [2]:
```
dfLegitimateUsers = pd.read_csv("E:/4545/legitimate_users.txt", sep="\t", enco
dfLegitimateUsers.columns = ["UserID", "CreatedAt", "CollectedAt", "NumberOfFo
dfContentPolluters = pd.read_csv("E:/4545/spammers.txt", sep="\t", encoding='u
dfContentPolluters.columns = ["UserID", "CreatedAt", "CollectedAt", "NumberOfF

dfLegitimateFollowings = pd.read_csv("E:/4545/legitimate_users_followings.txt"
dfLegitimateFollowings.columns = ["UserID", "SeriesOfNumberOfFollowings"]
dfPollutersFollowings = pd.read_csv("E:/4545/spammers_following.txt", sep="\t"
dfPollutersFollowings.columns = ["UserID", "SeriesOfNumberOfFollowings"]

dfLegitimateTweets = pd.read_csv("E:/4545/legitimate_users_tweets.txt", sep="\
dfLegitimateTweets.columns = ["UserID", "TweetID", "Tweet", "CreatedAt"]
dfPollutersTweets = pd.read_csv("E:/4545/spammers_tweets.txt", sep="\t", encod
dfPollutersTweets.columns = ["UserID", "TweetID", "Tweet", "CreatedAt"]
```

Feature Engineering

In [3]:
```
dfUsers = dfLegitimateUsers[["UserID", "LengthOfScreenName", "LengthOfDescript
dfPolluters = dfContentPolluters[["UserID", "LengthOfScreenName", "LengthOfDes
```

In [4]:
```
# Account Age

dfUsers["AccountAge(hours)"] = (pd.to_datetime(dfLegitimateUsers["CollectedAt"
dfPolluters["AccountAge(hours)"] = (pd.to_datetime(dfContentPolluters["Collect
```

In [5]:
```
# Follower Ratio

dfUsers["FollowingsOnFollowersRatio"] = dfUsers["NumberOfFollowings"] / dfUser
dfUsers.replace([np.inf, -np.inf], np.nan, inplace=True)
dfPolluters["FollowingsOnFollowersRatio"] = dfPolluters["NumberOfFollowings"]
dfPolluters.replace([np.inf, -np.inf], np.nan, inplace=True)
```

In [6]:
```
# Reputation Score

dfUsers["ReputationScore"] =  dfUsers["NumberOfFollowers"] / (dfUsers["NumberO
dfUsers.replace([np.inf, -np.inf], np.nan, inplace=True)
dfPolluters["ReputationScore"] =  dfPolluters["NumberOfFollowers"] / (dfPollut
dfPolluters.replace([np.inf, -np.inf], np.nan, inplace=True)
```

In [7]:
```
# Tweets On Account Age Ratio

dfUsers["TweetsOnLifetimeRatio"] = dfUsers["NumberOfTweets"] / dfUsers["Accoun
dfPolluters["TweetsOnLifetimeRatio"] = dfPolluters["NumberOfTweets"] / dfPollu
```

In [8]:
```
# Average Tweets Per Day

dfLegitimateTweets["CreatedAtDay"] = pd.to_datetime(dfLegitimateTweets["Create
dfLegitimateTweetsDay = dfLegitimateTweets.groupby(["UserID", "CreatedAtDay"])
dfLegitimateAvTweetsDay = dfLegitimateTweetsDay.groupby("UserID").mean().reset
dfLegitimateTweets.drop("CreatedAtDay", axis=1, inplace=True)
dfUsers = dfUsers.merge(dfLegitimateAvTweetsDay, how="outer")

dfPollutersTweets["CreatedAtDay"] = pd.to_datetime(dfPollutersTweets["CreatedA
dfPollutersTweetsDay = dfPollutersTweets.groupby(["UserID", "CreatedAtDay"]).s
dfPollutersAvTweetsDay = dfPollutersTweetsDay.groupby("UserID").mean().reset_i
dfPollutersTweets.drop("CreatedAtDay", axis=1, inplace=True)
dfPolluters = dfPolluters.merge(dfPollutersAvTweetsDay, how="outer")
```

In [9]:
```
# Links On Tweets Ratio
# Average Number Of Links Per Tweet
# '@' Signs On Tweets Ratio
dfLegitimateRatio = pd.DataFrame(dfLegitimateTweets["UserID"])
dfLegitimateRatio["LinksOnTweetsRatio"] = dfLegitimateTweets["Tweet"].str.coun
dfLegitimateRatio.loc[dfLegitimateRatio.LinksOnTweetsRatio > 1, "LinksOnTweets
dfLegitimateRatio["LinksAvTweets"] = dfLegitimateTweets["Tweet"].str.count("ht
dfLegitimateRatio["AtSignsOnTweetsRatio"] = dfLegitimateTweets["Tweet"].str.co
dfLegitimateRatio.loc[dfLegitimateRatio.AtSignsOnTweetsRatio > 1, "AtSignsOnTw

dfLegitimateLinkRatio = dfLegitimateRatio.groupby("UserID").sum()["LinksOnTwee
dfLegitimateLinkRatio.columns = ["UserID", "LinksOnTweetsRatio"]
dfLegitimateLinkAv = dfLegitimateRatio.groupby("UserID").mean()["LinksAvTweets
dfLegitimateLinkAv.columns = ["UserID", "AverageNumberOfLinksPerTweet"]
dfLegitimateAtSignRatio = dfLegitimateRatio.groupby("UserID").sum()["AtSignsOn
dfLegitimateAtSignRatio.columns = ["UserID", "AtSignsOnTweetsRatio"]
dfUsers = dfUsers.merge(dfLegitimateLinkRatio, left_on="UserID", right_on="Use
dfUsers["LinksOnTweetsRatio"] = dfUsers["LinksOnTweetsRatio"] / dfUsers["Numbe
dfUsers = dfUsers.merge(dfLegitimateLinkAv, left_on="UserID", right_on="UserID
dfUsers = dfUsers.merge(dfLegitimateAtSignRatio, left_on="UserID", right_on="U
dfUsers["AtSignsOnTweetsRatio"] = dfUsers["AtSignsOnTweetsRatio"] / dfUsers["N

dfPollutersRatio = pd.DataFrame(dfPollutersTweets["UserID"])
dfPollutersRatio["LinksOnTweetsRatio"] = dfPollutersTweets["Tweet"].str.count(
dfPollutersRatio.loc[dfPollutersRatio.LinksOnTweetsRatio > 1, "LinksOnTweetsRa
dfPollutersRatio["LinksAvTweets"] = dfPollutersTweets["Tweet"].str.count("http
dfPollutersRatio["AtSignsOnTweetsRatio"] = dfPollutersTweets["Tweet"].str.coun
dfPollutersRatio.loc[dfPollutersRatio.AtSignsOnTweetsRatio > 1, "AtSignsOnTwee

dfPollutersLinkRatio = dfPollutersRatio.groupby("UserID").sum()["LinksOnTweets
dfPollutersLinkRatio.columns = ["UserID", "LinksOnTweetsRatio"]
dfPollutersLinkAv = dfPollutersRatio.groupby("UserID").mean()["LinksAvTweets"]
dfPollutersLinkAv.columns = ["UserID", "AverageNumberOfLinksPerTweet"]
dfPollutersAtSignRatio = dfPollutersRatio.groupby("UserID").sum()["AtSignsOnTw
dfPollutersAtSignRatio.columns = ["UserID", "AtSignsOnTweetsRatio"]
dfPolluters = dfPolluters.merge(dfPollutersLinkRatio, left_on="UserID", right_
dfPolluters["LinksOnTweetsRatio"] = dfPolluters["LinksOnTweetsRatio"] / dfPoll
dfPolluters = dfPolluters.merge(dfPollutersLinkAv, left_on="UserID", right_on=
dfPolluters = dfPolluters.merge(dfPollutersAtSignRatio, left_on="UserID", righ
dfPolluters["AtSignsOnTweetsRatio"] = dfPolluters["AtSignsOnTweetsRatio"] / df
```

In [10]:
```
# Consecutive Tweets Average TimeDiff
# Consecutive Tweets Max     TimeDiff

dfLegitimateTweetsTimeDiff = pd.DataFrame(dfLegitimateTweets[["UserID", "Creat
dfLegitimateTweetsTimeDiff.sort_values(by=["UserID", "CreatedAt"], inplace=Tru
dfLegitimateTweetsTimeDiff["TweetsTimeDiff"] = pd.to_datetime(dfLegitimateTwee
dfLegitimateTweetsTimeDiff.drop("CreatedAt", axis=1, inplace=True)
dfLegitimateTweetsTimeDiff["TweetsTimeDiff"] = dfLegitimateTweetsTimeDiff.grou
dfLegitimateTweetsAvDiff = dfLegitimateTweetsTimeDiff.groupby("UserID").mean()
dfLegitimateTweetsAvDiff.columns = ["ConsecutiveTweetsAverageTimeDiff"]
dfLegitimateTweetsMaxDiff = dfLegitimateTweetsTimeDiff.groupby("UserID").max()
dfLegitimateTweetsMaxDiff.columns = ["ConsecutiveTweetsMaxTimeDiff"]
dfUsers = dfUsers.merge(dfLegitimateTweetsAvDiff, left_on="UserID", right_on="
dfUsers = dfUsers.merge(dfLegitimateTweetsMaxDiff, left_on="UserID", right_on=

dfPollutersTweetsTimeDiff = pd.DataFrame(dfPollutersTweets[["UserID", "Created
dfPollutersTweetsTimeDiff.sort_values(by=["UserID", "CreatedAt"], inplace=True
dfPollutersTweetsTimeDiff["TweetsTimeDiff"] = pd.to_datetime(dfPollutersTweets
dfPollutersTweetsTimeDiff.drop("CreatedAt", axis=1, inplace=True)
dfPollutersTweetsTimeDiff["TweetsTimeDiff"] = dfPollutersTweetsTimeDiff.groupb
dfPollutersTweetsAvDiff = dfPollutersTweetsTimeDiff.groupby("UserID").mean()
dfPollutersTweetsAvDiff.columns = ["ConsecutiveTweetsAverageTimeDiff"]
dfPollutersTweetsMaxDiff = dfPollutersTweetsTimeDiff.groupby("UserID").max()
dfPollutersTweetsMaxDiff.columns = ["ConsecutiveTweetsMaxTimeDiff"]
dfPolluters = dfPolluters.merge(dfPollutersTweetsAvDiff, left_on="UserID", rig
dfPolluters = dfPolluters.merge(dfPollutersTweetsMaxDiff, left_on="UserID", ri
```

In [11]:
```
# Average Tweet Length

dfLegitimateLength = pd.DataFrame(dfLegitimateTweets["UserID"])
dfLegitimateLength["AvgLengthOfTweet"] = dfLegitimateTweets["Tweet"].str.len()
dfLegitimateAvgLength = dfLegitimateLength.groupby("UserID").mean()["AvgLength
dfLegitimateAvgLength.columns = ["UserID", "AvgLengthOfTweet"]
dfUsers = dfUsers.merge(dfLegitimateAvgLength, left_on="UserID", right_on="Use

dfPollutersLength = pd.DataFrame(dfPollutersTweets["UserID"])
dfPollutersLength["AvgLengthOfTweet"] = dfPollutersTweets["Tweet"].str.len()
dfPollutersAvgLength = dfPollutersLength.groupby("UserID").mean()["AvgLengthOf
dfPollutersAvgLength.columns = ["UserID", "AvgLengthOfTweet"]
dfPolluters = dfPolluters.merge(dfPollutersAvgLength, left_on="UserID", right_
```

In [12]:
```
# No. Of Punctuations in Users

#No. Of HASHTAGS
dfLegitimateHashtags = pd.DataFrame(dfLegitimateTweets["UserID"])
dfLegitimateHashtags["HashtagsOnTweets"] = dfLegitimateTweets["Tweet"].str.cou
dfLegitimateTotalHashtags = dfLegitimateHashtags.groupby("UserID").sum()["Hash
dfLegitimateTotalHashtags.columns = ["UserID", "TotalHashtags"]
dfUsers = dfUsers.merge(dfLegitimateTotalHashtags, left_on="UserID", right_on=

#No. Of EMark
dfLegitimateEMark = pd.DataFrame(dfLegitimateTweets["UserID"])
dfLegitimateEMark["EMarkOnTweets"] = dfLegitimateTweets["Tweet"].str.count('!'
dfLegitimateTotalEMark = dfLegitimateEMark.groupby("UserID").sum()["EMarkOnTwe
dfLegitimateTotalEMark.columns = ["UserID", "TotalEMark"]
dfUsers = dfUsers.merge(dfLegitimateTotalEMark, left_on="UserID", right_on="Us

#No. Of Commas
dfLegitimateCommas = pd.DataFrame(dfLegitimateTweets["UserID"])
dfLegitimateCommas["CommasOnTweets"] = dfLegitimateTweets["Tweet"].str.count('
dfLegitimateTotalCommas = dfLegitimateCommas.groupby("UserID").sum()["CommasOn
dfLegitimateTotalCommas.columns = ["UserID", "TotalCommas"]
dfUsers = dfUsers.merge(dfLegitimateTotalCommas, left_on="UserID", right_on="U

#No. Of QMark
dfLegitimateQMark = pd.DataFrame(dfLegitimateTweets["UserID"])
q = []
for i in range(len(dfLegitimateTweets["Tweet"])):
    q.append(str(dfLegitimateTweets["Tweet"][i]).count('?'))
dfLegitimateQMark["QMarkOnTweets"] = q
dfLegitimateTotalQMark = dfLegitimateQMark.groupby("UserID").sum()["QMarkOnTwe
dfLegitimateTotalQMark.columns = ["UserID", "TotalQMark"]
dfUsers = dfUsers.merge(dfLegitimateTotalQMark, left_on="UserID", right_on="Us

#No. Of Parentheses
dfLegitimateParentheses = pd.DataFrame(dfLegitimateTweets["UserID"])
cp = []
for i in range(len(dfLegitimateTweets["Tweet"])):
    cp.append(str(dfLegitimateTweets["Tweet"][i]).count(')'))
op = []
for i in range(len(dfLegitimateTweets["Tweet"])):
    op.append(str(dfLegitimateTweets["Tweet"][i]).count('('))
dfLegitimateParentheses["ParenthesesOnTweets"] = list(map(add, cp, op))
dfLegitimateTotalParentheses = dfLegitimateParentheses.groupby("UserID").sum()
dfLegitimateTotalParentheses.columns = ["UserID", "TotalParentheses"]
dfUsers = dfUsers.merge(dfLegitimateTotalParentheses, left_on="UserID", right_

#No. of dots
dfLegitimateDots = pd.DataFrame(dfLegitimateTweets["UserID"])
d = []
for i in range(len(dfLegitimateTweets["Tweet"])):
    d.append(str(dfLegitimateTweets["Tweet"][i]).count('.'))
dfLegitimateDots["DotsOnTweets"] = d
dfLegitimateTotalDots = dfLegitimateDots.groupby("UserID").sum()["DotsOnTweets
dfLegitimateTotalDots.columns = ["UserID", "TotalDots"]
dfUsers = dfUsers.merge(dfLegitimateTotalDots, left_on="UserID", right_on="Use
```

```
#No. Of Dash
dfLegitimateDash = pd.DataFrame(dfLegitimateTweets["UserID"])
dfLegitimateDash["DashOnTweets"] = dfLegitimateTweets["Tweet"].str.count('-')
dfLegitimateTotalDash = dfLegitimateDash.groupby("UserID").sum()["DashOnTweets
dfLegitimateTotalDash.columns = ["UserID", "TotalDash"]
dfUsers = dfUsers.merge(dfLegitimateTotalDash, left_on="UserID", right_on="Use

#No. Of Hyphen
dfLegitimateHyphen = pd.DataFrame(dfLegitimateTweets["UserID"])
dfLegitimateHyphen["HyphenOnTweets"] = dfLegitimateTweets["Tweet"].str.count('
dfLegitimateTotalHyphen = dfLegitimateHyphen.groupby("UserID").sum()["HyphenOn
dfLegitimateTotalHyphen.columns = ["UserID", "TotalHyphen"]
dfUsers = dfUsers.merge(dfLegitimateTotalHyphen, left_on="UserID", right_on="U

#No. Of SemiColon
dfLegitimateSemiColon = pd.DataFrame(dfLegitimateTweets["UserID"])
dfLegitimateSemiColon["SemiColonOnTweets"] = dfLegitimateTweets["Tweet"].str.c
dfLegitimateTotalSemiColon = dfLegitimateSemiColon.groupby("UserID").sum()["Se
dfLegitimateTotalSemiColon.columns = ["UserID", "TotalSemiColon"]
dfUsers = dfUsers.merge(dfLegitimateTotalSemiColon, left_on="UserID", right_on

#No. Of Colon
dfLegitimateColon = pd.DataFrame(dfLegitimateTweets["UserID"])
dfLegitimateColon["ColonOnTweets"] = dfLegitimateTweets["Tweet"].str.count(';'
dfLegitimateTotalColon = dfLegitimateColon.groupby("UserID").sum()["ColonOnTwe
dfLegitimateTotalColon.columns = ["UserID", "TotalColon"]
dfUsers = dfUsers.merge(dfLegitimateTotalColon, left_on="UserID", right_on="Us

#No. Of Apostrophe
dfLegitimateApostrophe = pd.DataFrame(dfLegitimateTweets["UserID"])
dfLegitimateApostrophe["ApostropheOnTweets"] = dfLegitimateTweets["Tweet"].str
dfLegitimateTotalApostrophe = dfLegitimateApostrophe.groupby("UserID").sum()["
dfLegitimateTotalApostrophe.columns = ["UserID", "TotalApostrophe"]
dfUsers = dfUsers.merge(dfLegitimateTotalApostrophe, left_on="UserID", right_o

#No. Of Slash
dfLegitimateSlash = pd.DataFrame(dfLegitimateTweets["UserID"])
dfLegitimateSlash["SlashOnTweets"] = dfLegitimateTweets["Tweet"].str.count('/'
dfLegitimateTotalSlash = dfLegitimateSlash.groupby("UserID").sum()["SlashOnTwe
dfLegitimateTotalSlash.columns = ["UserID", "TotalSlash"]
dfUsers = dfUsers.merge(dfLegitimateTotalSlash, left_on="UserID", right_on="Us

#No. Of SQuotes
dfLegitimateSQuotes = pd.DataFrame(dfLegitimateTweets["UserID"])
dfLegitimateSQuotes["SQuotesOnTweets"] = dfLegitimateTweets["Tweet"].str.count
dfLegitimateTotalSQuotes = dfLegitimateSQuotes.groupby("UserID").sum()["SQuote
dfLegitimateTotalSQuotes.columns = ["UserID", "TotalSQuotes"]
dfUsers = dfUsers.merge(dfLegitimateTotalSQuotes, left_on="UserID", right_on="

#No. Of EQuotes
dfLegitimateEQuotes = pd.DataFrame(dfLegitimateTweets["UserID"])
dfLegitimateEQuotes["EQuotesOnTweets"] = dfLegitimateTweets["Tweet"].str.count
dfLegitimateTotalEQuotes = dfLegitimateEQuotes.groupby("UserID").sum()["EQuote
dfLegitimateTotalEQuotes.columns = ["UserID", "TotalEQuotes"]
dfUsers = dfUsers.merge(dfLegitimateTotalEQuotes, left_on="UserID", right_on="
```

```
dfUsers["TotalPunctuations"] = dfUsers["TotalHashtags"] + dfUsers["TotalEMark"]
dfUsers = dfUsers.drop(['TotalHashtags', 'TotalEMark', 'TotalCommas','TotalQMa
```

In [13]:
```
# No. Of Punctuations in Polluters

#No. Of HASHTAGS
dfPollutersHashtags = pd.DataFrame(dfPollutersTweets["UserID"])
dfPollutersHashtags["HashtagsOnTweets"] = dfPollutersTweets["Tweet"].str.count
dfPollutersTotalHashtags = dfPollutersHashtags.groupby("UserID").sum()["Hashta
dfPollutersTotalHashtags.columns = ["UserID", "TotalHashtags"]
dfPolluters = dfPolluters.merge(dfPollutersTotalHashtags, left_on="UserID", ri

#No. Of EMark
dfPollutersEMark = pd.DataFrame(dfPollutersTweets["UserID"])
dfPollutersEMark["EMarkOnTweets"] = dfPollutersTweets["Tweet"].str.count('!')
dfPollutersTotalEMark = dfPollutersEMark.groupby("UserID").sum()["EMarkOnTweet
dfPollutersTotalEMark.columns = ["UserID", "TotalEMark"]
dfPolluters = dfPolluters.merge(dfPollutersTotalEMark, left_on="UserID", right

#No. Of Commas
dfPollutersCommas = pd.DataFrame(dfPollutersTweets["UserID"])
dfPollutersCommas["CommasOnTweets"] = dfPollutersTweets["Tweet"].str.count(','
dfPollutersTotalCommas = dfPollutersCommas.groupby("UserID").sum()["CommasOnTw
dfPollutersTotalCommas.columns = ["UserID", "TotalCommas"]
dfPolluters = dfPolluters.merge(dfPollutersTotalCommas, left_on="UserID", righ

#No. Of QMark
dfPollutersQMark = pd.DataFrame(dfPollutersTweets["UserID"])
q = []
for i in range(len(dfPollutersTweets["Tweet"])):
    q.append(str(dfPollutersTweets["Tweet"][i]).count('?'))
dfPollutersQMark["QMarkOnTweets"] = q
dfPollutersTotalQMark = dfPollutersQMark.groupby("UserID").sum()["QMarkOnTweet
dfPollutersTotalQMark.columns = ["UserID", "TotalQMark"]
dfPolluters = dfPolluters.merge(dfPollutersTotalQMark, left_on="UserID", right

#No. Of Parentheses
dfPollutersParentheses = pd.DataFrame(dfPollutersTweets["UserID"])
cp = []
for i in range(len(dfPollutersTweets["Tweet"])):
    cp.append(str(dfPollutersTweets["Tweet"][i]).count(')'))
op = []
for i in range(len(dfPollutersTweets["Tweet"])):
    op.append(str(dfPollutersTweets["Tweet"][i]).count('('))
dfPollutersParentheses["ParenthesesOnTweets"] = list(map(add, cp, op))
dfPollutersTotalParentheses = dfPollutersParentheses.groupby("UserID").sum()["
dfPollutersTotalParentheses.columns = ["UserID", "TotalParentheses"]
dfPolluters = dfPolluters.merge(dfPollutersTotalParentheses, left_on="UserID",

#No. of dots
dfPollutersDots = pd.DataFrame(dfPollutersTweets["UserID"])
d = []
for i in range(len(dfPollutersTweets["Tweet"])):
    d.append(str(dfPollutersTweets["Tweet"][i]).count('.'))
dfPollutersDots["DotsOnTweets"] = d
dfPollutersTotalDots = dfPollutersDots.groupby("UserID").sum()["DotsOnTweets"]
dfPollutersTotalDots.columns = ["UserID", "TotalDots"]
dfPolluters = dfPolluters.merge(dfPollutersTotalDots, left_on="UserID", right_
```

```
#No. Of Dash
dfPollutersDash = pd.DataFrame(dfPollutersTweets["UserID"])
dfPollutersDash["DashOnTweets"] = dfPollutersTweets["Tweet"].str.count('—')
dfPollutersTotalDash = dfPollutersDash.groupby("UserID").sum()["DashOnTweets"]
dfPollutersTotalDash.columns = ["UserID", "TotalDash"]
dfPolluters = dfPolluters.merge(dfPollutersTotalDash, left_on="UserID", right_

#No. Of Hyphen
dfPollutersHyphen = pd.DataFrame(dfPollutersTweets["UserID"])
dfPollutersHyphen["HyphenOnTweets"] = dfPollutersTweets["Tweet"].str.count('-'
dfPollutersTotalHyphen = dfPollutersHyphen.groupby("UserID").sum()["HyphenOnTw
dfPollutersTotalHyphen.columns = ["UserID", "TotalHyphen"]
dfPolluters = dfPolluters.merge(dfPollutersTotalHyphen, left_on="UserID", righ

#No. Of SemiColon
dfPollutersSemiColon = pd.DataFrame(dfPollutersTweets["UserID"])
dfPollutersSemiColon["SemiColonOnTweets"] = dfPollutersTweets["Tweet"].str.cou
dfPollutersTotalSemiColon = dfPollutersSemiColon.groupby("UserID").sum()["Semi
dfPollutersTotalSemiColon.columns = ["UserID", "TotalSemiColon"]
dfPolluters = dfPolluters.merge(dfPollutersTotalSemiColon, left_on="UserID", r

#No. Of Colon
dfPollutersColon = pd.DataFrame(dfPollutersTweets["UserID"])
dfPollutersColon["ColonOnTweets"] = dfPollutersTweets["Tweet"].str.count(';')
dfPollutersTotalColon = dfPollutersColon.groupby("UserID").sum()["ColonOnTweet
dfPollutersTotalColon.columns = ["UserID", "TotalColon"]
dfPolluters = dfPolluters.merge(dfPollutersTotalColon, left_on="UserID", right

#No. Of Apostrophe
dfPollutersApostrophe = pd.DataFrame(dfPollutersTweets["UserID"])
dfPollutersApostrophe["ApostropheOnTweets"] = dfPollutersTweets["Tweet"].str.c
dfPollutersTotalApostrophe = dfPollutersApostrophe.groupby("UserID").sum()["Ap
dfPollutersTotalApostrophe.columns = ["UserID", "TotalApostrophe"]
dfPolluters = dfPolluters.merge(dfPollutersTotalApostrophe, left_on="UserID",

#No. Of Slash
dfPollutersSlash = pd.DataFrame(dfPollutersTweets["UserID"])
dfPollutersSlash["SlashOnTweets"] = dfPollutersTweets["Tweet"].str.count('/')
dfPollutersTotalSlash = dfPollutersSlash.groupby("UserID").sum()["SlashOnTweet
dfPollutersTotalSlash.columns = ["UserID", "TotalSlash"]
dfPolluters = dfPolluters.merge(dfPollutersTotalSlash, left_on="UserID", right

#No. Of SQuotes
dfPollutersSQuotes = pd.DataFrame(dfPollutersTweets["UserID"])
dfPollutersSQuotes["SQuotesOnTweets"] = dfPollutersTweets["Tweet"].str.count('
dfPollutersTotalSQuotes = dfPollutersSQuotes.groupby("UserID").sum()["SQuotesO
dfPollutersTotalSQuotes.columns = ["UserID", "TotalSQuotes"]
dfPolluters = dfPolluters.merge(dfPollutersTotalSQuotes, left_on="UserID", rig

#No. Of EQuotes
dfPollutersEQuotes = pd.DataFrame(dfPollutersTweets["UserID"])
dfPollutersEQuotes["EQuotesOnTweets"] = dfPollutersTweets["Tweet"].str.count('
dfPollutersTotalEQuotes = dfPollutersEQuotes.groupby("UserID").sum()["EQuotesO
dfPollutersTotalEQuotes.columns = ["UserID", "TotalEQuotes"]
dfPolluters = dfPolluters.merge(dfPollutersTotalEQuotes, left_on="UserID", rig
```

```
dfPolluters["TotalPunctuations"] = dfPolluters["TotalHashtags"] + dfPolluters[
dfPolluters = dfPolluters.drop(['TotalHashtags', 'TotalEMark', 'TotalCommas','
```

In [14]:
```
dfUsers.head()
```

Out[14]:

	UserID	LengthOfScreenName	LengthOfDescriptionInUserProfile	NumberOfFollowings	Number
0	614	10	34	510	
1	1038	7	156	304	
2	1437	6	37	45	
3	2615	7	0	211	
4	3148	8	97	7346	

In [15]:
```
dfUsers.shape
```

Out[15]: (350, 18)

In [16]:
```
dfPolluters.head()
```

Out[16]:

	UserID	LengthOfScreenName	LengthOfDescriptionInUserProfile	NumberOfFollowings	Number
0	6301	8	132	3269	
1	10836	9	134	1949	
2	10997	12	158	1119	
3	633293	11	121	2174	
4	717883	6	70	7731	

In [17]:
```
dfPolluters.shape
```

Out[17]: (341, 18)

In [18]:
```
dfUsers.to_csv("dfUsers.csv", encoding='utf-8', index=False)
```

In [19]:
```
dfPolluters.to_csv("dfPolluters.csv", encoding='utf-8', index=False)
```

In [20]:
```
dfUsers = pd.read_csv('dfUsers.csv')
dfPolluters = pd.read_csv('dfPolluters.csv')
```

Text Preprocessing

In [21]:
```
import nltk
#nltk.download('stopwords')
#nltk.download('wordnet')
import pandas as pd
from nltk.corpus import stopwords
import re
import string
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.decomposition import TruncatedSVD
import spacy, gensim
from sklearn.decomposition import LatentDirichletAllocation
from pprint import pprint
```

In [22]:
```
dfLegitimateTweet = pd.DataFrame(dfLegitimateTweets["UserID"])
dfLegitimateTweet["Tweets"] = dfLegitimateTweets["Tweet"]
dfLegitimateAllTweet = dfLegitimateTweet.groupby("UserID").sum()["Tweets"].res
dfLegitimateAllTweet.columns = ["UserID", "TotalJoin"]
```

In [23]:
```
dfPollutersTweet = pd.DataFrame(dfPollutersTweets["UserID"])
dfPollutersTweet["Tweets"] = dfPollutersTweets["Tweet"]
dfPollutersAllTweet = dfPollutersTweet.groupby("UserID").sum()["Tweets"].reset
dfPollutersAllTweet.columns = ["UserID", "TotalJoin"]
```

In [24]:
```
dfLegitimateAllTweet.shape
```

Out[24]: (349, 2)

In [25]:
```
dfPollutersAllTweet.shape
```

Out[25]: (341, 2)

Removing URLs, Punctuations, Numbers, Underscores & Lowercase-ing

In [26]:
```
dfLegitimateAllTweet["TotalJoin"] = dfLegitimateAllTweet["TotalJoin"].replace(
dfLegitimateAllTweet["TotalJoin"] = dfLegitimateAllTweet["TotalJoin"].replace(
dfLegitimateAllTweet["TotalJoin"] = dfLegitimateAllTweet["TotalJoin"].replace(
dfLegitimateAllTweet["TotalJoin"] = dfLegitimateAllTweet["TotalJoin"].str.repl
dfLegitimateAllTweet["TotalJoin"] = dfLegitimateAllTweet["TotalJoin"].str.repl
dfLegitimateAllTweet["TotalJoin"] = dfLegitimateAllTweet["TotalJoin"].str.lowe
for i in range(len(dfLegitimateAllTweet["TotalJoin"])):
    t=[w for w in str(dfLegitimateAllTweet["TotalJoin"][i]).split() if len(w)<
    dfLegitimateAllTweet["TotalJoin"][i] = " ".join(t)
```

In [27]:
```
dfPollutersAllTweet["TotalJoin"] = dfPollutersAllTweet["TotalJoin"].replace(r'
dfPollutersAllTweet["TotalJoin"] = dfPollutersAllTweet["TotalJoin"].replace(r'
dfPollutersAllTweet["TotalJoin"] = dfPollutersAllTweet["TotalJoin"].replace(r'
dfPollutersAllTweet["TotalJoin"] = dfPollutersAllTweet["TotalJoin"].str.replac
dfPollutersAllTweet["TotalJoin"] = dfPollutersAllTweet["TotalJoin"].str.replac
dfPollutersAllTweet["TotalJoin"] = dfPollutersAllTweet["TotalJoin"].str.lower(
for i in range(len(dfPollutersAllTweet["TotalJoin"])):
    t=[w for w in str(dfPollutersAllTweet["TotalJoin"][i]).split() if len(w)<1
    dfPollutersAllTweet["TotalJoin"][i] = " ".join(t)
```

In [37]:
```
import nltk
nltk.download('wordnet')
```

```
[nltk_data] Downloading package wordnet to C:\Users\Kaustubh
[nltk_data]     Yewale\AppData\Roaming\nltk_data...
```

Out[37]:
```
True
```

Removing Stopwords

In [33]:
```
stop=set(stopwords.words("english"))

for i in range(len(dfLegitimateAllTweet["TotalJoin"])):
    t=[word for word in str(dfLegitimateAllTweet["TotalJoin"][i]).split() if w
    dfLegitimateAllTweet["TotalJoin"][i] = " ".join(t)

for i in range(len(dfPollutersAllTweet["TotalJoin"])):
    t=[word for word in str(dfPollutersAllTweet["TotalJoin"][i]).split() if wo
    dfPollutersAllTweet["TotalJoin"][i] = " ".join(t)
```

Spelling Checker

In [35]:
```
from spellchecker import SpellChecker
spell = SpellChecker(distance=1)
def Correct(x):
    return spell.correction(x)
dfLegitimateAllTweet["TotalJoin"] = dfLegitimateAllTweet["TotalJoin"].apply(Co
dfPollutersAllTweet["TotalJoin"] = dfPollutersAllTweet["TotalJoin"].apply(Corr
```

Lemmatizing

In [38]:
```
lemmatizer = WordNetLemmatizer()
for i in range(len(dfLegitimateAllTweet["TotalJoin"])):
    t=[lemmatizer.lemmatize(word) for word in str(dfLegitimateAllTweet["TotalJ
    dfLegitimateAllTweet["TotalJoin"][i] = " ".join(t)
for i in range(len(dfPollutersAllTweet["TotalJoin"])):
    t=[lemmatizer.lemmatize(word) for word in str(dfPollutersAllTweet["TotalJo
    dfPollutersAllTweet["TotalJoin"][i] = " ".join(t)
```

In [39]:
```
dfLegitimateAllTweet.to_csv("dfLegitimateAllTweet.csv", encoding='utf-8', inde
dfPollutersAllTweet.to_csv("dfPollutersAllTweet.csv", encoding='utf-8', index=
```

In [40]:
```
dfLegitimateAllTweet = pd.read_csv('dfLegitimateAllTweet.csv')
dfPollutersAllTweet = pd.read_csv('dfPollutersAllTweet.csv')
```

TF-IDF

In [41]:
```
v = TfidfVectorizer()
u = v.fit_transform(dfLegitimateAllTweet["TotalJoin"])
p = v.fit_transform(dfPollutersAllTweet["TotalJoin"])
```

In [42]:
```
u.shape
```

Out[42]:
```
(349, 74961)
```

In [43]:
```
p.shape
```

Out[43]:
```
(341, 51958)
```

In [44]:
```
#u.toarray()
```

generally when decomposition of this kind is done on text data, the terms SVD and LSA are used interchangeably

In [45]:
```
svd = TruncatedSVD()
svd.fit(u)
Userstransformed = svd.transform(u)
svd.fit(p)
Polluterstransformed = svd.transform(p)
```

In [46]:
```
Ulsa = pd.DataFrame(Userstransformed)
Ulsa.columns =['LSA1', 'LSA2']
Ulsa
```

Out[46]:

	LSA1	LSA2
0	0.210042	-0.252529
1	0.240486	0.130782
2	0.281432	-0.332973
3	0.227658	-0.265186
4	0.254880	-0.284355
...	...	...
344	0.221989	-0.248264
345	0.443970	0.357024
346	0.310797	-0.376997
347	0.232985	-0.265131
348	0.249532	-0.305001

349 rows × 2 columns

In [47]:
```
Plsa = pd.DataFrame(Polluterstransformed)
Plsa.columns =['LSA1', 'LSA2']
Plsa
```

Out[47]:

	LSA1	LSA2
0	0.118266	-0.051937
1	0.118171	0.024870
2	0.341092	-0.100422
3	0.351612	-0.130452
4	0.130166	-0.035628
...	...	...
336	0.244668	0.112783
337	0.225187	0.038841
338	0.442934	-0.026526
339	0.472637	-0.194547
340	0.319834	-0.110269

341 rows × 2 columns

In [48]:
```
#dfUsers['fulltweet'] = dfLegitimateAllTweet["TotalJoin"]
dfUsers = dfUsers.merge(dfLegitimateAllTweet, left_on="UserID", right_on="User
dfPolluters = dfPolluters.merge(dfPollutersAllTweet, left_on="UserID", right_o
```

In [49]:
```
dfUsers = dfUsers[dfUsers['TotalJoin'].notna()]
```

In [50]:
```
dfPolluters = dfPolluters[dfPolluters['TotalJoin'].notna()]
```

In [51]:
```
dfUsers.shape
```

Out[51]: (349, 19)

In [52]:
```
dfPolluters.shape
```

Out[52]: (341, 19)

In [53]:
```
dfUsers.reset_index(drop=True, inplace=True)
```

In [54]:
```
dfPolluters.reset_index(drop=True, inplace=True)
```

In [55]:
```
dfUsers
```

Out[55]:

	UserID	LengthOfScreenName	LengthOfDescriptionInUserProfile	NumberOfFollowings	Num
0	614	10	34	510	
1	1038	7	156	304	
2	1437	6	37	45	
3	2615	7	0	211	
4	3148	8	97	7346	
...	...	...	...	...	
344	5931162	9	137	1835	
345	5937312	7	158	9524	
346	5942122	8	0	111	

	UserID	LengthOfScreenName	LengthOfDescriptionInUserProfile	NumberOfFollowings	Num
347	5945472	10	54	156	
348	5947912	7	32	52	

In [56]:
```
dfUsers = pd.concat([dfUsers,Ulsa], axis=1)
```

In [57]:
```
dfPolluters = pd.concat([dfPolluters,Plsa], axis=1)
```

LDA

In [58]:
```
vectorizer = CountVectorizer(analyzer='word',
                             min_df=10,                          # minimum requi
                             stop_words='english',               # remove stop w
                             lowercase=True,                     # convert all w
                             token_pattern='[a-zA-Z0-9]{3,}',    # num chars > 3
                             # max_features=50000,               # max number of
                            )
data_vectorized = vectorizer.fit_transform(dfLegitimateAllTweet["TotalJoin"])
```

In [59]:
```
# Build LDA Model
lda_model = LatentDirichletAllocation(n_components=20,               # Number
                                      max_iter=10,                   # Max lea
                                      learning_method='online',
                                      random_state=100,              # Random
                                      batch_size=128,                # n docs
                                      evaluate_every = -1,           # compute
                                      n_jobs = -1,                   # Use all
                                     )
Ulda_output = lda_model.fit_transform(data_vectorized)
print(lda_model)
```

```
LatentDirichletAllocation(learning_method='online', n_components=20, n_jobs=-
1,
                          random_state=100)
```

In [60]:
```
Ulda_output.shape
```

Out[60]:
```
(349, 20)
```

In [61]:
```
Ulda = pd.DataFrame(Ulda_output)
Ulda.columns =['LDA1', 'LDA2', 'LDA3', 'LDA4', 'LDA5', 'LDA6', 'LDA7', 'LDA8',
Ulda
```

Out[61]:

	LDA1	LDA2	LDA3	LDA4	LDA5	LDA6	LDA7	LDA8	LDA9	
0	0.000084	0.000084	0.000084	0.000084	0.000084	0.976228	0.000084	0.000084	0.000084	0
1	0.000141	0.000141	0.000141	0.000141	0.000141	0.000141	0.000141	0.000141	0.000141	0
2	0.000065	0.000065	0.000065	0.000065	0.000065	0.998758	0.000065	0.000065	0.000065	0
3	0.000076	0.000076	0.000076	0.000076	0.000076	0.852316	0.000076	0.000076	0.000076	0
4	0.000075	0.000075	0.000075	0.000075	0.000075	0.039579	0.000075	0.000075	0.000075	0
...	...	...	...	...	...	...	...	...	...	
344	0.000057	0.000057	0.000057	0.000057	0.000057	0.000057	0.000057	0.000057	0.000057	0
345	0.000087	0.000087	0.000087	0.000087	0.000087	0.000087	0.000087	0.000087	0.998342	0
346	0.000060	0.000060	0.000060	0.000060	0.000060	0.998851	0.000060	0.000060	0.000060	0
347	0.000061	0.000061	0.000061	0.000061	0.000061	0.672790	0.000061	0.000061	0.000061	0
348	0.000065	0.000065	0.000065	0.000065	0.000065	0.998769	0.000065	0.000065	0.000065	0

349 rows × 20 columns

In [62]:
```
dfUsers = pd.concat([dfUsers,Ulda], axis=1)
```

In [63]:
```
vectorizer = CountVectorizer(analyzer='word',
                             min_df=10,                          # minimum requi
                             stop_words='english',               # remove stop w
                             lowercase=True,                     # convert all w
                             token_pattern='[a-zA-Z0-9]{3,}',    # num chars > 3
                             # max_features=50000,               # max number of
                            )
data_vectorized = vectorizer.fit_transform(dfPollutersAllTweet["TotalJoin"])
```

In [64]:
```
# Build LDA Model
lda_model = LatentDirichletAllocation(n_components=20,               # Number
                                      max_iter=10,                   # Max lea
                                      learning_method='online',
                                      random_state=100,              # Random
                                      batch_size=128,                # n docs
                                      evaluate_every = -1,           # compute
                                      n_jobs = -1,                   # Use all
                                     )
Plda_output = lda_model.fit_transform(data_vectorized)
print(lda_model)
```

```
LatentDirichletAllocation(learning_method='online', n_components=20, n_jobs=-
1,
                          random_state=100)
```

In [65]:
```
Plda_output.shape
```

Out[65]: (341, 20)

In [66]:
```
Plda = pd.DataFrame(Plda_output)
Plda.columns =['LDA1', 'LDA2', 'LDA3', 'LDA4', 'LDA5', 'LDA6', 'LDA7', 'LDA8',
Plda
```

Out[66]:

	LDA1	LDA2	LDA3	LDA4	LDA5	LDA6	LDA7	LDA8	LDA9	
0	0.000065	0.011060	0.000065	0.000065	0.000065	0.000065	0.000065	0.987770	0.000065	0
1	0.171135	0.000092	0.043653	0.442896	0.000092	0.000092	0.000092	0.340837	0.000092	0
2	0.000053	0.000053	0.014404	0.032953	0.000053	0.000053	0.000053	0.862035	0.000053	0
3	0.000065	0.000065	0.000065	0.063849	0.000065	0.000065	0.000065	0.892506	0.000065	0
4	0.000049	0.000049	0.000049	0.000049	0.000049	0.000049	0.000049	0.000049	0.000049	0
...	...	...	...	...	...	...	...	...	...	
336	0.000202	0.000202	0.019967	0.025848	0.000202	0.056920	0.000202	0.268117	0.000202	0
337	0.000223	0.000223	0.000223	0.367156	0.000223	0.000223	0.000223	0.185632	0.000223	0
338	0.000043	0.000043	0.000043	0.000043	0.000043	0.000043	0.000043	0.571456	0.000043	0
339	0.000045	0.000045	0.000045	0.000045	0.000045	0.000045	0.000045	0.958054	0.000045	0
340	0.000071	0.000071	0.000071	0.000071	0.000071	0.000071	0.000071	0.931531	0.000071	0

341 rows × 20 columns

In [67]:
```
dfPolluters = pd.concat([dfPolluters,Plda], axis=1)
```

In [68]:
```
dfUsers.drop(['TotalJoin'], axis = 1, inplace = True)
```

In [69]:
```
dfPolluters.drop(['TotalJoin'], axis = 1, inplace = True)
```

In [70]:
```
dfUsers["Class"] = 0
dfPolluters["Class"] = 1
df = dfUsers.append(dfPolluters)
df.to_csv("FinalSHP.csv", encoding='utf-8', index=False)
```

In [71]:
```
df.shape
```

Out[71]: (690, 41)

In [72]:
```
df =  pd.read_csv('FinalSHP.csv')
```

In [73]:
```
df.head()
```

Out[73]:

	UserID	LengthOfScreenName	LengthOfDescriptionInUserProfile	NumberOfFollowings	Number(
0	614	10	34	510	
1	1038	7	156	304	
2	1437	6	37	45	
3	2615	7	0	211	
4	3148	8	97	7346	

5 rows × 41 columns

In [74]:
```
df.columns
```

Out[74]:
```
Index(['UserID', 'LengthOfScreenName', 'LengthOfDescriptionInUserProfile',
       'NumberOfFollowings', 'NumberOfFollowers', 'NumberOfTweets',
       'AccountAge(hours)', 'FollowingsOnFollowersRatio', 'ReputationScore',
       'TweetsOnLifetimeRatio', 'AverageTweetsPerDay', 'LinksOnTweetsRatio',
       'AverageNumberOfLinksPerTweet', 'AtSignsOnTweetsRatio',
       'ConsecutiveTweetsAverageTimeDiff', 'ConsecutiveTweetsMaxTimeDiff',
       'AvgLengthOfTweet', 'TotalPunctuations', 'LSA1', 'LSA2', 'LDA1', 'LDA2
',
       'LDA3', 'LDA4', 'LDA5', 'LDA6', 'LDA7', 'LDA8', 'LDA9', 'LDA10',
       'LDA11', 'LDA12', 'LDA13', 'LDA14', 'LDA15', 'LDA16', 'LDA17', 'LDA18
',
       'LDA19', 'LDA20', 'Class'],
      dtype='object')
```

Handling Null values

In [75]:
```
df.isnull().sum()
```

Out[75]:
```
UserID                                  0
LengthOfScreenName                      0
LengthOfDescriptionInUserProfile        0
NumberOfFollowings                      0
NumberOfFollowers                       0
NumberOfTweets                          0
AccountAge(hours)                       0
FollowingsOnFollowersRatio              1
ReputationScore                         1
TweetsOnLifetimeRatio                   0
AverageTweetsPerDay                     0
LinksOnTweetsRatio                      0
AverageNumberOfLinksPerTweet            0
AtSignsOnTweetsRatio                    0
ConsecutiveTweetsAverageTimeDiff        0
ConsecutiveTweetsMaxTimeDiff            0
AvgLengthOfTweet                        0
TotalPunctuations                       0
LSA1                                    0
LSA2                                    0
LDA1                                    0
LDA2                                    0
LDA3                                    0
LDA4                                    0
LDA5                                    0
LDA6                                    0
LDA7                                    0
LDA8                                    0
LDA9                                    0
LDA10                                   0
LDA11                                   0
LDA12                                   0
LDA13                                   0
LDA14                                   0
LDA15                                   0
LDA16                                   0
LDA17                                   0
LDA18                                   0
LDA19                                   0
LDA20                                   0
Class                                   0
dtype: int64
```

In [78]:
```
help(df.dropna())
```

```
Help on DataFrame in module pandas.core.frame object:

class DataFrame(pandas.core.generic.NDFrame, pandas.core.arraylike.OpsMixin)
 |  DataFrame(data=None, index: 'Axes | None' = None, columns: 'Axes | None'
= None, dtype: 'Dtype | None' = None, copy: 'bool | None' = None)
 |
 |  Two-dimensional, size-mutable, potentially heterogeneous tabular data.
 |
 |  Data structure also contains labeled axes (rows and columns).
 |  Arithmetic operations align on both row and column labels. Can be
 |  thought of as a dict-like container for Series objects. The primary
 |  pandas data structure.
 |
 |  Parameters
 |  ----------
 |  data : ndarray (structured or homogeneous), Iterable, dict, or DataFrame
 |      Dict can contain Series, arrays, constants, dataclass or list-like ob
jects. If
 |      data is a dict, column order follows insertion-order.
 '
```

In [79]:
```
df.dropna(how='any',inplace=True)
```

In [80]:
```
df.isnull().sum()
```

Out[80]:
```
UserID                                0
LengthOfScreenName                    0
LengthOfDescriptionInUserProfile      0
NumberOfFollowings                    0
NumberOfFollowers                     0
NumberOfTweets                        0
AccountAge(hours)                     0
FollowingsOnFollowersRatio            0
ReputationScore                       0
TweetsOnLifetimeRatio                 0
AverageTweetsPerDay                   0
LinksOnTweetsRatio                    0
AverageNumberOfLinksPerTweet          0
AtSignsOnTweetsRatio                  0
ConsecutiveTweetsAverageTimeDiff      0
ConsecutiveTweetsMaxTimeDiff          0
AvgLengthOfTweet                      0
TotalPunctuations                     0
LSA1                                  0
LSA2                                  0
LDA1                                  0
LDA2                                  0
LDA3                                  0
LDA4                                  0
LDA5                                  0
LDA6                                  0
LDA7                                  0
LDA8                                  0
LDA9                                  0
LDA10                                 0
LDA11                                 0
LDA12                                 0
LDA13                                 0
LDA14                                 0
LDA15                                 0
LDA16                                 0
LDA17                                 0
LDA18                                 0
LDA19                                 0
LDA20                                 0
Class                                 0
dtype: int64
```

In [81]:
```
df.reset_index(drop=True, inplace=True)
```

In [82]:
```
df.shape
```

Out[82]:
```
(689, 41)
```

In [83]:
```
X = df.drop('Class', axis=1)
y = df['Class']
```

In [84]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
```

In [85]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, ran
```

Normalization

In [86]:
```
sc = StandardScaler()
X_train = sc.fit_transform(X_train.astype(float))
X_test = sc.fit_transform(X_test.astype(float))
```

Naive Bayes

In [87]:
```
nb = GaussianNB()
```

In [88]:
```
nb.fit(X_train, y_train)
```

Out[88]:
```
▾ GaussianNB

GaussianNB()
```

In [89]:
```
y_pred = nb.predict(X_test)
accuracy_score(y_test, y_pred)
```

Out[89]:
```
0.9130434782608695
```

J48

In [90]:
```
dte = DecisionTreeClassifier(criterion = "entropy", max_depth = 3, min_samples
```

In [91]:
```
dte.fit(X_train, y_train)
```

Out[91]:
```
▾                              DecisionTreeClassifier

DecisionTreeClassifier(criterion='entropy', max_depth=3, min_samples_leaf=5)
```

In [92]:
```
y_pred = dte.predict(X_test)
accuracy_score(y_test, y_pred)
```

Out[92]:
```
0.8695652173913043
```

SVM

In [93]:
```
svc = SVC(gamma='auto')
```

In [94]:
```
svc.fit(X_train, y_train)
```

Out[94]:
```
SVC(gamma='auto')
```

In [95]:
```
y_pred = svc.predict(X_test)
accuracy_score(y_test, y_pred)
```

Out[95]:
```
0.9565217391304348
```

XGBOOST

In [96]:
```
gb = GradientBoostingClassifier()
```

In [97]:
```
gb.fit(X_train, y_train)
```

Out[97]:
```
GradientBoostingClassifier()
```

In [98]:
```
y_pred = gb.predict(X_test)
accuracy_score(y_test, y_pred)
```

Out[98]:
```
0.8695652173913043
```

MLP

In [99]:
```
mlp = MLPClassifier(random_state=1)
```

In [100]:
```
mlp.fit(X_train, y_train)
```

Out[100]:
```
MLPClassifier(random_state=1)
```

In [101]:
```
y_pred = mlp.predict(X_test)
accuracy_score(y_test, y_pred)
```

Out[101]:
```
0.9758454106280193
```

Random Forest

In [102]:
```
rfc = RandomForestClassifier()
```

In [103]:
```
rfc.fit(X_train, y_train)
```

Out[103]:
```
▾ RandomForestClassifier

RandomForestClassifier()
```

In [104]:
```
y_pred = rfc.predict(X_test)
accuracy_score(y_test, y_pred)
```

Out[104]:
```
0.9371980676328503
```

In [ ]:
```

```