

# A Comparative Study of Machine Learning Algorithms for the Prediction of Drug-Abuse - A Classification Approach

January 1, 2024

## 1 Ali Ahmad Siddiqui

### 1.1 Contra Costa College

### 1.2 Dr. Chao Liu

### 1.3 Fall 2023

## 2 Table of Contents

- 3. Abstract
- 4. Introduction
- 5. Goals
- 6. Literature Review
- 7. Dataset Information
- 8. Methodology
- 9. Dataset Preprocessing
- 10. Correlation Analysis
- 11. Feature Selection
- 12. Model Training and Learning
- 13. Results & Discussion
- 14. Conclusion & Future Plans
- 15. References

## 3 Abstract

This research was conducted as part of ENGIN-298 at Contra Costa College by Ali Ahmad Siddiqui with the assistance of Dr. Chao Liu from Fall 2022 to Fall 2023. Drug-use is a societal crisis that is in-need of desperate mitigation and alleviation. In this paper, we claim that machine learning algorithms are highly effective in assessing the likelihood for an individual to engage in the use of certain drugs. In particular, we show how machine learning algorithms can be trained using real-world data collected from users who consume a variety of different drugs in order to give insight into the drug-use for unknown individuals.

## 4 Introduction

Substance abuse in the United States remains an unsolved epidemic. In 2022, there has been an 29% increase in substance abuse (compared to 2019), with over 99,000+ Americans killed due to substance overdose just in 2020 alone. Drug overdose deaths in the US since 2000 are nearing one million. And even though the federal budget for drug control nears \$35 billion dollars, there are many, unexplored approaches to mitigate the current crisis [4]. Recently, in the realm of programming, machine learning has become a powerful computational tool that serves to identify patterns in large data sets. While research has been conducted on machine learning in the context of drug-abuse, little has been done on using specific and quantifiable personality factors as a means of training these assessing the risk of drug-abuse.

## 5 Goals

Machine Learning has the potential to improve drug-abuse diagnosis by providing a non-invasive and efficient method in the process of assessing an individual's risk prediction. Our research aims to identify accurate models and improve the accuracy of these models to identify high-risk individuals early on. In the process, our research help contributes to to early interventions and reducing the monetary and physical burden of the criminal justice and healthcare systems. However, it is imperative to note to that this research is not a substitute for diagnosis and the final conclusions should always be made by industry professionals, including therapists, counselors, and psychiatrist.

As a result, the purpose of this study is to examine the effectiveness of machine learning algorithms to assess the likelihood of specific drug-uses for an individual using personality risk-factors. The importance of this is to identify those at-risk at an earlier state to ensure they seek the proper medical and counseling resources to ensure such interests don't turn into dangerous addictions. The objective of this independent study is to identify machine learning programs in the Python language that can accurately predict the likelihood for abuse of specific drugs and substances for an individual, as a means of alleviating the suffering that society faces due to drug addiction and overdoses.

## 6 Literature Review

In 2020, over 40 million people in America had at least one substance use disorder. Substance use disorder is used to describe the recurring pattern of using a substance that may cause problems, distress, or even death [3].

Substance abuse not only affects individual drug users, but also their families, friends, and society at large, and are rapidly consuming limited public funds. The economic impact of drug abuse is significant to the economy at large because of lost productivity, burglary, violence, assault, and theft. In 2002, the economic cost of drug abuse to the United States was \$180.9 billion. Children of individuals who abuse drugs are often abused or neglected [5]. According to the Federal Bureau of Prisons, around 46% of inmates were incarcerated for drug offenses or drug-related crimes in 2021 [2].

Artificial intelligence systems, with the help of Deep Learning and Machine Learning, can serve as tools to help the current suffering across the nation by finding a way to identify substance abuse in its early phases to ensure it does not exacerbate. Machine learning is a branch of computer science that uses data to overcome challenges that seem impossible through traditional methods.

This is done by allowing computers to learn without directly programming that learning. For example, machine learning has been used to expand drug development prior to critical shortages, especially in the case of infection surges during globalized pandemics [1]. Using Machine Learning to understand risk factors associated with drug-abuse can better enhance counseling and medical intervention services by providing a tool to identify individuals on a risky-road early on.

## 7 Dataset Information

For the project, I used a free, publicly available machine learning repository from UC Irvine [7]. The data available in the repository used an online survey methodology to collect data, which includes the Big Five personality traits (NEO-FFI-R), impulsivity (BIS-11), sensation seeking (ImpSS), and demographic information. The purpose of this specificc dataset is to diagnostically predict the level of specific drug-use an individual , based on certain personality and impulsivity factors and measurements available in the dataset for each individual. The database contains records for 1885 respondents. For each respondent 12 attributes are known: Personality measurements which include NEO-FFI-R (neuroticism, extraversion, openness to experience, agreeableness, and conscientiousness), BIS-11 (impulsivity), and ImpSS (sensation seeking), level of education, age, gender, country of residence and ethnicity.

All input attributes are originally categorical and are quantified. After quantification values of all input features can be considered as real-valued. In addition, participants were questioned concerning their use of 18 legal and illegal drugs (alcohol, amphetamines, amyl nitrite, benzodiazepine, cannabis, chocolate, cocaine, caffeine, crack, ecstasy, heroin, ketamine, legal highs, LSD, methadone, mushrooms, nicotine and volatile substance abuse and one fictitious drug (Semeron) which was introduced to identify over-claimers.

For each drug they have to select one of the answers: never used the drug, used it over a decade ago, or in the last decade, year, month, week, or day.

1. ID is number of record in original database. Cannot be related to participant. It can be used for reference only.
2. Age (Real) is age of participant and has one of the values:

Value	Meaning	Cases	Fraction
-0.95197	18-24	643	34.11%
-0.07854	25-34	481	25.52%
0.49788	35-44	356	18.89%
1.09449	45-54	294	15.60%
1.82213	55-64	93	4.93%
2.59171	65+	18	0.95%

Min	Max	Mean	Std.dev.
-0.95197	2.59171	0.03461	0.87813

3. Gender (Real) is gender of participant:

Value	Meaning	Cases	Fraction
0.48246	Female	942	49.97%
-0.48246	Male	943	50.03%

Min	Max	Mean	Std.dev.
-0.48246	0.48246	-0.00026	0.48246

4. Education (Real) is level of education of participant and has one of the values:

Value	Meaning	Cases	Fraction
-2.43591	Left school before 16 years	28	1.49%
-1.73790	Left school at 16 years	99	5.25%
-1.43719	Left school at 17 years	30	1.59%
-1.22751	Left school at 18 years	100	5.31%
-0.61113	Some college or university, no certificate or degree	506	26.84%
-0.05921	Professional certificate/ diploma	270	14.32%
0.45468	University degree	480	25.46%
1.16365	Masters degree	283	15.01%
1.98437	Doctorate degree	89	4.72%

Min	Max	Mean	Std.dev.
-2.43591	1.98437	-0.00379	0.95004

5. Country (Real) is country of current residence of participant and has one of the values:

Value	Meaning	Cases	Fraction
-0.09765	Australia	54	2.86%
0.24923	Canada	87	4.62%
-0.46841	New Zealand	5	0.27%
-0.28519	Other	118	6.26%
0.21128	Republic of Ireland	20	1.06%
0.96082	UK	1044	55.38%
-0.57009	USA	557	29.55%

Min	Max	Mean	Std.dev.
-0.57009	0.96082	0.35554	0.70015

6. Ethnicity (Real) is ethnicity of participant and has one of the values:

Value	Meaning	Cases	Fraction
-0.50212	Asian	26	1.38%
-1.10702	Black	33	1.75%
1.90725	Mixed-Black/Asian	3	0.16%
0.12600	Mixed-White/Asian	20	1.06%
-0.22166	Mixed-White/Black	20	1.06%
0.11440	Other	63	3.34%
-0.31685	White	1720	91.25%

Min	Max	Mean	Std.dev.
-1.10702	1.90725	-0.30958	0.16618

7. Nscore (Real) is NEO-FFI-R Neuroticism. Possible values are presented in table below:

Nscore	Cases	Value	Nscore	Cases	Value	Nscore	Cases	Value
12	1	-3.46436	29	60	-0.67825	46	67	1.02119
13	1	-3.15735	30	61	-0.58016	47	27	1.13281
14	7	-2.75696	31	87	-0.46725	48	49	1.23461
15	4	-2.52197	32	78	-0.34799	49	40	1.37297
16	3	-2.42317	33	68	-0.24649	50	24	1.49158
17	4	-2.34360	34	76	-0.14882	51	27	1.60383
18	10	-2.21844	35	69	-0.05188	52	17	1.72012
19	16	-2.05048	36	73	0.04257	53	20	1.83990
20	24	-1.86962	37	67	0.13606	54	15	1.98437
21	31	-1.69163	38	63	0.22393	55	11	2.12700
22	26	-1.55078	39	66	0.31287	56	10	2.28554
23	29	-1.43907	40	80	0.41667	57	6	2.46262
24	35	-1.32828	41	61	0.52135	58	3	2.61139
25	56	-1.19430	42	77	0.62967	59	5	2.82196
26	57	-1.05308	43	49	0.73545	60	2	3.27393
27	65	-0.92104	44	51	0.82562			
28	70	-0.79151	45	37	0.91093			

Min	Max	Mean	Std.dev.
-3.46436	3.27393	0.00004	0.99808

8. Escore (Real) is NEO-FFI-R Extraversion. Possible values are presented in table below:

Escore	Cases	Value	Escore	Cases	Value	Escore	Cases	Value
16	2	-3.27393	31	55	-1.23177	45	91	0.80523
18	1	-3.00537	32	52	-1.09207	46	69	0.96248
19	6	-2.72827	33	77	-0.94779	47	64	1.11406

Escore	Cases	Value	Escore	Cases	Value	Escore	Cases	Value
20	3	-2.53830	34	68	-0.80615	48	62	1.28610
21	3	-2.44904	35	58	-0.69509	49	37	1.45421
22	8	-2.32338	36	89	-0.57545	50	25	1.58487
23	5	-2.21069	37	90	-0.43999	51	34	1.74091
24	9	-2.11437	38	106	-0.30033	52	21	1.93886
25	4	-2.03972	39	107	-0.15487	53	15	2.12700
26	21	-1.92173	40	130	0.00332	54	10	2.32338
27	23	-1.76250	41	116	0.16767	55	9	2.57309
28	23	-1.63340	42	109	0.32197	56	2	2.85950
29	32	-1.50796	43	105	0.47617	58	1	3.00537
30	38	-1.37639	44	103	0.63779	59	2	3.27393

Min	Max	Mean	Std.dev.
-3.27393	3.27393	-0.00016	0.99745

9. Oscore (Real) is NEO-FFI-R Openness to experience. Possible values are presented in table below:

Oscore	Cases	Value	Oscore	Cases	Value	Oscore	Cases	Value
24	2	-3.27393	38	64	-1.11902	50	83	0.58331
26	4	-2.85950	39	60	-0.97631	51	87	0.72330
28	4	-2.63199	40	68	-0.84732	52	87	0.88309
29	11	-2.39883	41	76	-0.71727	53	81	1.06238
30	9	-2.21069	42	87	-0.58331	54	57	1.24033
31	9	-2.09015	43	86	-0.45174	55	63	1.43533
32	13	-1.97495	44	101	-0.31776	56	38	1.65653
33	23	-1.82919	45	103	-0.17779	57	34	1.88511
34	25	-1.68062	46	134	-0.01928	58	19	2.15324
35	26	-1.55521	47	107	0.14143	59	13	2.44904
36	39	-1.42424	48	116	0.29338	60	7	2.90161
37	51	-1.27553	49	98	0.44585			

Min	Max	Mean	Std.dev.
-3.27393	2.90161	-0.00053	0.99623

10. Ascore (Real) is NEO-FFI-R Agreeableness. Possible values are presented in table below:

Ascore	Cases	Value	Ascore	Cases	Value	Ascore	Cases	Value
12	1	-3.46436	34	42	-1.34289	48	104	0.76096
16	1	-3.15735	35	45	-1.21213	49	85	0.94156
18	1	-3.00537	36	62	-1.07533	50	68	1.11406
23	1	-2.90161	37	83	-0.91699	51	58	1.2861

Ascore	Cases	Value	Ascore	Cases	Value	Ascore	Cases	Value
24	2	-2.78793	38	82	-0.76096	52	39	1.45039
25	1	-2.70172	39	102	-0.60633	53	36	1.61108
26	7	-2.5383	40	98	-0.45321	54	36	1.81866
27	7	-2.35413	41	114	-0.30172	55	16	2.03972
28	8	-2.21844	42	101	-0.15487	56	14	2.23427
29	13	-2.07848	43	105	-0.01729	57	8	2.46262
30	18	-1.92595	44	118	0.13136	58	7	2.75696
31	24	-1.772	45	112	0.28783	59	1	3.15735
32	30	-1.6209	46	100	0.43852	60	1	3.46436
33	34	-1.47955	47	100	0.59042			

Min	Max	Mean	Std.dev.
-3.46436	3.46436	-0.00024	0.99744

11. Cscore (Real) is NEO-FFI-R Conscientiousness. Possible values are presented in table below:

Cscore	Cases	Value	Cscore	Cases	Value	Cscore	Cases	Value
17	1	-3.46436	32	39	-1.25773	46	113	0.58489
19	1	-3.15735	33	49	-1.13788	47	95	0.7583
20	3	-2.90161	34	55	-1.0145	48	95	0.93949
21	2	-2.72827	35	55	-0.89891	49	76	1.13407
22	5	-2.57309	36	69	-0.78155	50	47	1.30612
23	5	-2.42317	37	81	-0.65253	51	43	1.46191
24	6	-2.30408	38	77	-0.52745	52	34	1.63088
25	9	-2.18109	39	87	-0.40581	53	28	1.81175
26	13	-2.04506	40	97	-0.27607	54	27	2.04506
27	13	-1.92173	41	99	-0.14277	55	13	2.33337
28	25	-1.78169	42	105	-0.00665	56	8	2.63199
29	24	-1.64101	43	90	0.12331	57	3	3.00537
30	29	-1.5184	44	111	0.25953	59	1	3.46436
31	41	-1.38502	45	111	0.41594			

Min	Max	Mean	Std.dev.
-3.46436	3.46436	-0.00039	0.99752

12. Impulsive (Real) is impulsiveness measured by BIS-11. Possible values are presented in table below:

Impulsiveness	Cases	Fraction
-2.55524	20	1.06%
-1.37983	276	14.64%
-0.71126	307	16.29%

Impulsiveness	Cases	Fraction
-0.21712	355	18.83%
0.19268	257	13.63%
0.52975	216	11.46%
0.88113	195	10.34%
1.29221	148	7.85%
1.86203	104	5.52%
2.90161	7	0.37%

Min	Max	Mean	Std.dev.
-2.55524	2.90161	0.00721	0.95446

13. SS (Real) is sensation seeing measured by ImpSS. Possible values are presented in table below:

SS	Cases	Fraction
-2.07848	71	3.77%
-1.54858	87	4.62%
-1.18084	132	7.00%
-0.84637	169	8.97%
-0.52593	211	11.19%
-0.21575	223	11.83%
0.07987	219	11.62%
0.40148	249	13.21%
0.76540	211	11.19%
1.22470	210	11.14%
1.92173	103	5.46%

Min	Max	Mean	Std.dev.
-2.07848	1.92173	-0.00329	0.96370

14. Alcohol is class of alcohol consumption. It is output attribute with following distribution of classes.
15. Amphet is class of amphetamines consumption. It is output attribute with following distribution of classes.
16. Amyl is class of amyl nitrite consumption. It is output attribute with following distribution of classes.
17. Benzos is class of benzodiazepine consumption. It is output attribute with following distribution of classes:



Value	Class
CL0	Never Used
CL1	Used over a Decade Ago
CL2	Used in Last Decade
CL3	Used in Last Year
CL4	Used in Last Month
CL5	Used in Last Week
CL6	Used in Last Day

18. Caff is class of caffeine consumption. It is output attribute with following distribution of classes.
19. Cannabis is class of cannabis consumption. It is output attribute with following distribution of classes.
20. Choc is class of chocolate consumption. It is output attribute with following distribution of classes.
21. Coke is class of cocaine consumption. It is output attribute with following distribution of classes:

Value	Class
CL0	Never Used
CL1	Used over a Decade Ago
CL2	Used in Last Decade
CL3	Used in Last Year
CL4	Used in Last Month
CL5	Used in Last Week
CL6	Used in Last Day

22. Crack is class of crack consumption. It is output attribute with following distribution of classes.
23. Ecstasy is class of ecstasy consumption. It is output attribute with following distribution of classes.
24. Heroin is class of heroin consumption. It is output attribute with following distribution of classes.
25. Ketamine is class of ketamine consumption. It is output attribute with following distribution of classes:

Value	Class
CL0	Never Used
CL1	Used over a Decade Ago
CL2	Used in Last Decade
CL3	Used in Last Year
CL4	Used in Last Month
CL5	Used in Last Week

Value	Class
CL6	Used in Last Day

26. Legalh is class of legal highs consumption. It is output attribute with following distribution of classes
27. LSD is class of alcohol consumption. It is output attribute with following distribution of classes
28. Meth is class of methadone consumption. It is output attribute with following distribution of classes.
29. Mushrooms is class of magic mushrooms consumption. It is output attribute with following distribution of classes:

Value	Class
CL0	Never Used
CL1	Used over a Decade Ago
CL2	Used in Last Decade
CL3	Used in Last Year
CL4	Used in Last Month
CL5	Used in Last Week
CL6	Used in Last Day

30. Nicotine is class of nicotine consumption. It is output attribute with following distribution of classes.
31. Semer is class of fictitious drug Semeron consumption. It is output attribute with following distribution of classes.
32. VSA is class of volatile substance abuse consumption. It is output attribute with following distribution of classes:

Value	Class
CL0	Never Used
CL1	Used over a Decade Ago
CL2	Used in Last Decade
CL3	Used in Last Year
CL4	Used in Last Month
CL5	Used in Last Week
CL6	Used in Last Day

## 8 Methodology

### 8.1 Data Collection and Preprocessing

Preprocess the data available from the data repository. Handle missing values and ensure each row has real values to use for the project.

## 8.2 Correlation Analysis, Feature Selection

Conduct correlation analysis using the Python language among the variables in order to identify relationships between features and drug abuse. Select the features that display significant correlation and use that as the X feature data.

## 8.3 Dataset Preparation

Split the dataset into feature data (X) and target variable (Y), which will be used to create drug-use and abuse predictions in the machine learning models. Ensure the data is formatted properly for training.

## 8.4 Select Algorithms and Train

Choose a range of machine learning algorithms that can be trained on all 1885 respondents to predict drug-abuse for specific drugs. Use the trained models to generate predictions for drug use for each specific drug based on personality risk factors available from the dataset.

## 8.5 Model Evaluation, Model Selection

Evaluate the accuracy of each machine learning model's predictions against the actual data from the dataset. To evaluate the performance of the models, use train-test split to divide the dataset into training and testing set, specifically a 70%-training to 30%-testing ratio. Assess accuracy, precision, recall, and F1 score. Focus on the performance of models where correlations were established between features and drug use for the algorithms.

Evaluate the effectiveness of each model's predictions using cross-validation with 5 folds. Use the results from the cross-validation analysis to evaluate the effectiveness of the model for a specific drug

Select the model that demonstrates the highest combination of AUC score and F1 score in its predictions to strike balance, ensuring it can differentiate and discriminate between non-user and user, and can identify positive instances of drug-use for a specific drug without many false positives.

## 8.6 Assessment Metrics

Assess the final select model's performance using additional evaluation metrics. This includes precision, recall, and F1-score to ensure its reliability. Assess the AUC Score of the model for the specific drug using cross-validation.

## 8.7 Interpretation and Conclusion

Interpret the outcomes, and draw conclusions regarding effective models and their predictions for certain drugs.

# 9 Dataset Preprocessing

[ ]:

```
[49]: import pandas as pd
from sklearn.model_selection import train_test_split, cross_val_score,
↳StratifiedKFold
from sklearn.ensemble import RandomForestClassifier,
↳GradientBoostingClassifier, ExtraTreesClassifier
from sklearn.metrics import accuracy_score, classification_report,
↳confusion_matrix
from sklearn.linear_model import LogisticRegression
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC, LinearSVC, NuSVC
from sklearn.neighbors import KNeighborsClassifier, RadiusNeighborsClassifier,
↳NearestCentroid
from sklearn.neural_network import MLPClassifier
from pandas import read_csv
from pandas.plotting import scatter_matrix
from matplotlib import pyplot as plt
from scipy.stats import pearsonr
import seaborn as sns

# Load the dataset
dataset = pd.read_csv('data.csv')
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1885 entries, 0 to 1884
Data columns (total 32 columns):
#   Column      Non-Null Count  Dtype
---  -
0   ID           1885 non-null   int64
1   AGE          1885 non-null   float64
2   GENDER       1885 non-null   float64
3   EDUCATION    1885 non-null   float64
4   COUNTRY      1885 non-null   float64
5   ETHNICITY    1885 non-null   float64
6   N-SCORE      1885 non-null   float64
7   E-SCORE      1885 non-null   float64
8   OSCORE       1885 non-null   float64
9   ASCORE       1885 non-null   float64
10  CSCORE       1885 non-null   float64
11  IMPULSIVE    1885 non-null   float64
12  SS           1885 non-null   float64
13  ALCOHOL      1885 non-null   object
14  AMPHET       1885 non-null   object
15  AMYL         1885 non-null   object
16  BENZOS       1885 non-null   object
```

```

17 CAFF      1885 non-null  object
18 CANNABIS  1885 non-null  object
19 CHOC      1885 non-null  object
20 COKE      1885 non-null  object
21 CRACK     1885 non-null  object
22 ECSTASY   1885 non-null  object
23 HEROIN    1885 non-null  object
24 KETAMINE  1885 non-null  object
25 LEGALH    1885 non-null  object
26 LSD       1885 non-null  object
27 METH      1885 non-null  object
28 MUSHROOMS 1885 non-null  object
29 NICOTINE  1885 non-null  object
30 SEMER     1885 non-null  object
31 VSA       1885 non-null  object
dtypes: float64(12), int64(1), object(19)
memory usage: 471.4+ KB

```

```
[51]: #First 5 rows of the dataset
dataset.head()
```

```
[51]:
```

	ID	AGE	GENDER	EDUCATION	COUNTRY	ETHNICITY	N-SCORE	E-SCORE	\
0	1	0.49788	0.48246	-0.05921	0.96082	0.12600	0.31287	-0.57545	
1	2	-0.07854	-0.48246	1.98437	0.96082	-0.31685	-0.67825	1.93886	
2	3	0.49788	-0.48246	-0.05921	0.96082	-0.31685	-0.46725	0.80523	
3	4	-0.95197	0.48246	1.16365	0.96082	-0.31685	-0.14882	-0.80615	
4	5	0.49788	0.48246	1.98437	0.96082	-0.31685	0.73545	-1.63340	

	OSCORE	ASCORE	...	ECSTASY	HEROIN	KETAMINE	LEGALH	LSD	METH	\
0	-0.58331	-0.91699	...	CL0	CL0	CL0	CL0	CL0	CL0	
1	1.43533	0.76096	...	CL4	CL0	CL2	CL0	CL2	CL3	
2	-0.84732	-1.62090	...	CL0	CL0	CL0	CL0	CL0	CL0	
3	-0.01928	0.59042	...	CL0	CL0	CL2	CL0	CL0	CL0	
4	-0.45174	-0.30172	...	CL1	CL0	CL0	CL1	CL0	CL0	

	MUSHROOMS	NICOTINE	SEMER	VSA
0	CL0	CL2	CL0	CL0
1	CL0	CL4	CL0	CL0
2	CL1	CL0	CL0	CL0
3	CL0	CL2	CL0	CL0
4	CL2	CL2	CL0	CL0

```
[5 rows x 32 columns]
```

```
[53]: dataset.describe()
```

```
[53]:
```

	ID	AGE	GENDER	EDUCATION	COUNTRY \
count	1885.000000	1885.00000	1885.000000	1885.000000	1885.000000
mean	945.294960	0.03461	-0.000256	-0.003806	0.355542
std	545.167641	0.87836	0.482588	0.950078	0.700335
min	1.000000	-0.95197	-0.482460	-2.435910	-0.570090
25%	474.000000	-0.95197	-0.482460	-0.611130	-0.570090
50%	946.000000	-0.07854	-0.482460	-0.059210	0.960820
75%	1417.000000	0.49788	0.482460	0.454680	0.960820
max	1888.000000	2.59171	0.482460	1.984370	0.960820

	ETHNICITY	N-SCORE	E-SCORE	OSCORE	ASCORE \
count	1885.000000	1885.000000	1885.000000	1885.000000	1885.000000
mean	-0.309577	0.000047	-0.000163	-0.000534	-0.000245
std	0.166226	0.998106	0.997448	0.996229	0.997440
min	-1.107020	-3.464360	-3.273930	-3.273930	-3.464360
25%	-0.316850	-0.678250	-0.695090	-0.717270	-0.606330
50%	-0.316850	0.042570	0.003320	-0.019280	-0.017290
75%	-0.316850	0.629670	0.637790	0.723300	0.760960
max	1.907250	3.273930	3.273930	2.901610	3.464360

	CSCORE	IMPULSIVE	SS
count	1885.000000	1885.000000	1885.000000
mean	-0.000386	0.007216	-0.003292
std	0.997523	0.954435	0.963701
min	-3.464360	-2.555240	-2.078480
25%	-0.652530	-0.711260	-0.525930
50%	-0.006650	-0.217120	0.079870
75%	0.584890	0.529750	0.765400
max	3.464360	2.901610	1.921730

```
[55]: #Check if any column or row contain a null value
dataset.isna().sum()
```

```
[55]: ID          0
      AGE          0
      GENDER       0
      EDUCATION    0
      COUNTRY      0
      ETHNICITY     0
      N-SCORE       0
      E-SCORE       0
      OSCORE        0
      ASCORE        0
      CSCORE        0
      IMPULSIVE     0
      SS           0
      ALCOHOL       0
```

```

AMPHET      0
AMYL        0
BENZOS      0
CAFF        0
CANNABIS    0
CHOC        0
COKE        0
CRACK       0
ECSTASY     0
HEROIN      0
KETAMINE    0
LEGALH      0
LSD         0
METH        0
MUSHROOMS   0
NICOTINE    0
SEMER       0
VSA         0
dtype: int64

```

```
[57]: dataset.columns
```

```
[57]: Index(['ID', 'AGE', 'GENDER', 'EDUCATION', 'COUNTRY', 'ETHNICITY', 'N-SCORE',
          'E-SCORE', 'OSCORE', 'AScore', 'CSCORE', 'IMPULSIVE', 'SS', 'ALCOHOL',
          'AMPHET', 'AMYL', 'BENZOS', 'CAFF', 'CANNABIS', 'CHOC', 'COKE', 'CRACK',
          'ECSTASY', 'HEROIN', 'KETAMINE', 'LEGALH', 'LSD', 'METH', 'MUSHROOMS',
          'NICOTINE', 'SEMER', 'VSA'],
          dtype='object')
```

```
[61]: print(dataset.shape)
```

```
(1885, 32)
```

```
[65]: #Returns the number of unique values for each variable.
dataset.nunique()
```

```
[65]: ID          1885
      AGE          6
      GENDER       2
      EDUCATION    9
      COUNTRY      7
      ETHNICITY    7
      N-SCORE     49
      E-SCORE     42
      OSCORE      35
      AScore      41
      CSCORE      41
```

IMPULSIVE	10
SS	11
ALCOHOL	7
AMPHET	7
AMYL	7
BENZOS	7
CAFF	7
CANNABIS	7
CHOC	7
COKE	7
CRACK	7
ECSTASY	7
HEROIN	7
KETAMINE	7
LEGALH	7
LSD	7
METH	7
MUSHROOMS	7
NICOTINE	7
SEMER	5
VSA	7

dtype: int64

```
[72]: # Display the Missing Values
print(dataset.isna().sum())
```

ID	0
AGE	0
GENDER	0
EDUCATION	0
COUNTRY	0
ETHNICITY	0
N-SCORE	0
E-SCORE	0
OSCORE	0
AScore	0
CSCORE	0
IMPULSIVE	0
SS	0
ALCOHOL	0
AMPHET	0
AMYL	0
BENZOS	0
CAFF	0
CANNABIS	0
CHOC	0
COKE	0



CRACK	0
ECSTASY	0
HEROIN	0
KETAMINE	0
LEGALH	0
LSD	0
METH	0
MUSHROOMS	0
NICOTINE	0
SEMER	0
VSA	0

dtype: int64

## 10 Correlation Analysis

```
[107]: import matplotlib.pyplot as plt
#import the necessary libraries
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
#set the background style sheet
sns.set_style("whitegrid")

columns = ['ALCOHOL', 'AMPHET', 'AMYL', 'BENZOS', 'CAFF', 'CANNABIS', 'CHOC',
↳ 'COKE', 'CRACK', 'ECSTASY', 'HEROIN', 'KETAMINE', 'LEGALH', 'LSD', 'METH',
↳ 'MUSHROOMS', 'NICOTINE', 'SEMER', 'VSA']
cp = ['User_Alcohol', 'User_Amphet', 'User_Amyl', 'User_Benzos', 'User_Caff',
↳ 'User_Cannabis', 'User_Choc', 'User_Coke', 'User_Crack',
↳ 'User_Ecstasy', 'User_Heroin', 'User_Ketamine', 'User_Legalh',
↳ 'User_LSD', 'User_Meth', 'User_Mushrooms', 'User_Nicotine', 'User_Semer',
↳ 'User_VSA']

df = pd.read_csv('data.csv')
copy_df = df.copy() #make a copy of the dataframe

# Encode columns into numeric data
from sklearn.preprocessing import LabelEncoder
for column in columns:
    le = LabelEncoder()
    df[column] = le.fit_transform(df[column])

for column in columns:
    le = LabelEncoder()
    df[column] = le.fit_transform(df[column])
```

```

for column in columns:
    le = LabelEncoder()
    copy_df[column] = le.fit_transform(copy_df[column])

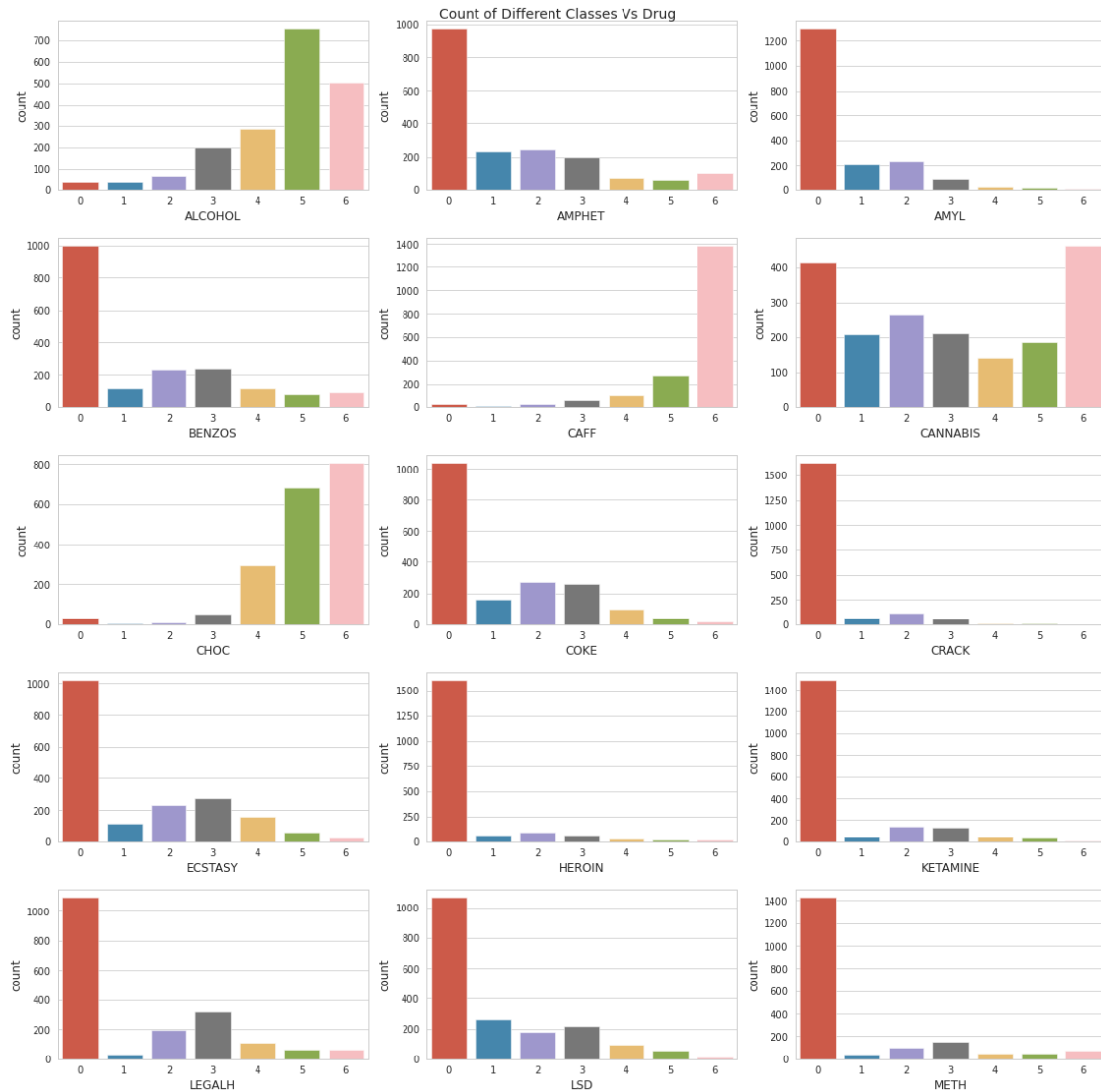
#make a new column for each drug which contain the information that a user is
↳using that drug or not
for i in range(len(columns)):
    copy_df.loc[((copy_df[columns[i]]==0) | (copy_df[columns[i]]==1)),cp[i]] =
    ↳'Non-user'
    copy_df.loc[((copy_df[columns[i]]==2) | (copy_df[columns[i]]==3) |
    ↳(copy_df[columns[i]]==4) | (copy_df[columns[i]]==5) |
    ↳(copy_df[columns[i]]==6)),cp[i]] = 'User'

fig, axes = plt.subplots(5,3,figsize = (16,16))
fig.suptitle("Count of Different Classes Vs Drug",fontsize=14)
k=0
for i in range(5):
    for j in range(3):
        sns.countplot(x=columns[k], data=copy_df,ax=axes[i][j])
        k+=1

plt.tight_layout()

plt.show()

```



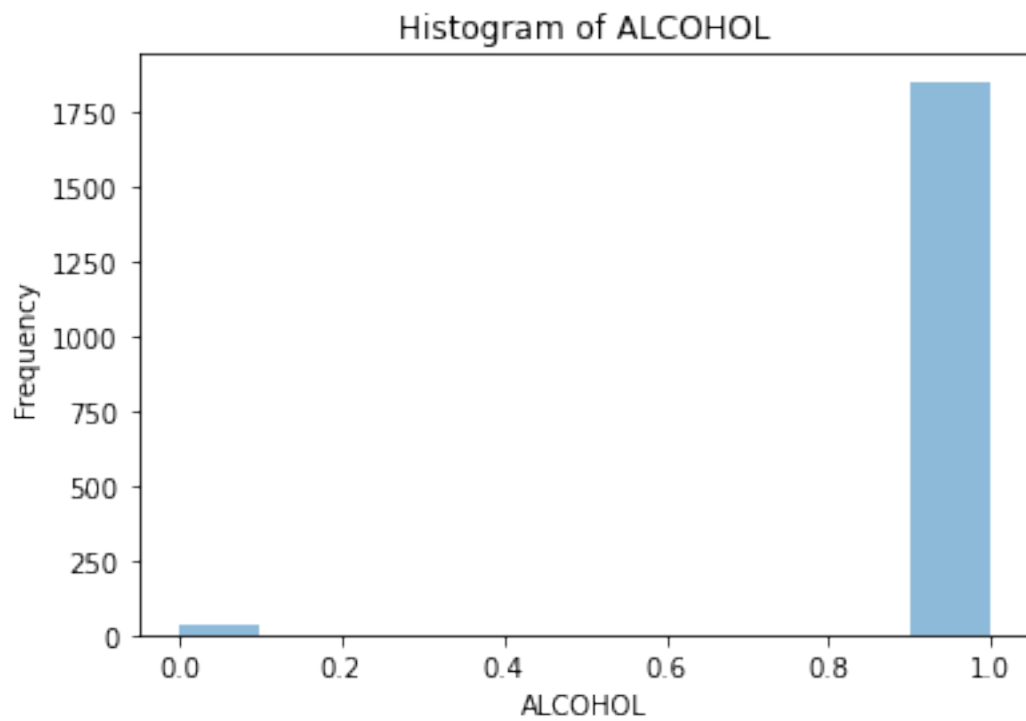
10.1 To ensure we have a single-class classification problem for this project, we are now using data6.csv, which converted the 7 classes of drug use into “Non-user” and User”. This data adjustment will now be used for the entirety of this project.

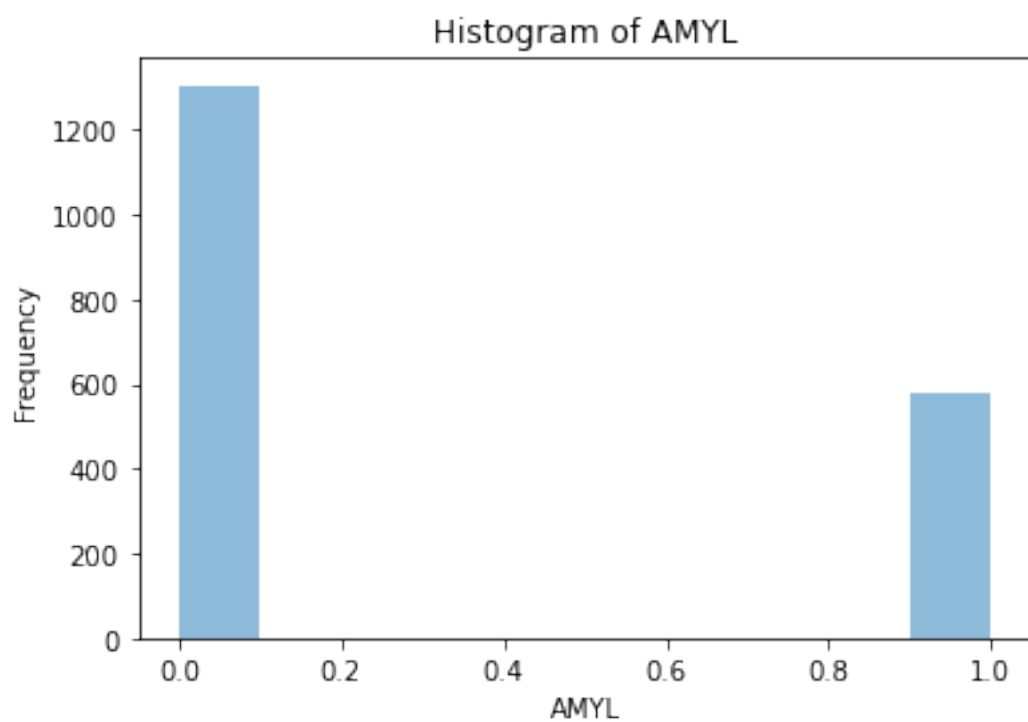
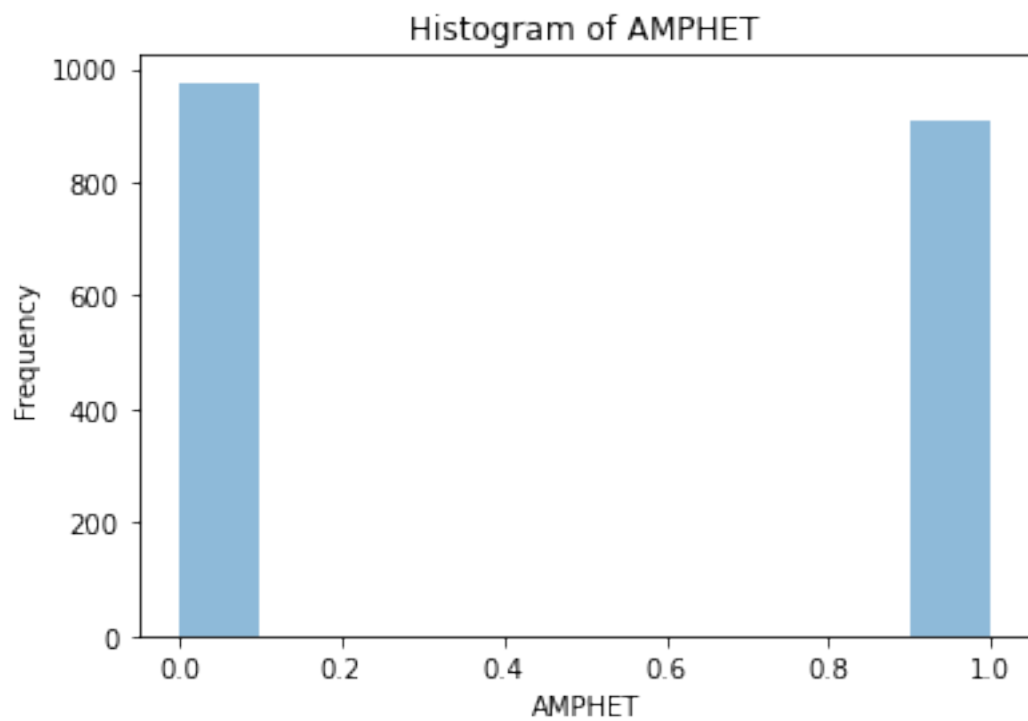
```
[167]: import pandas as pd
import matplotlib.pyplot as plt

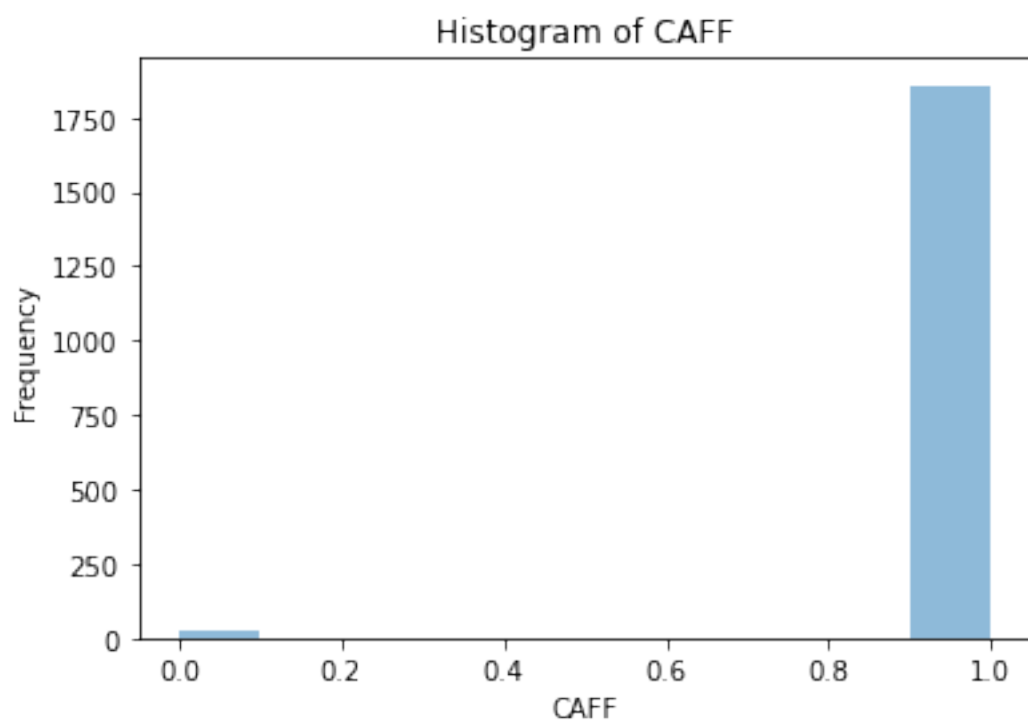
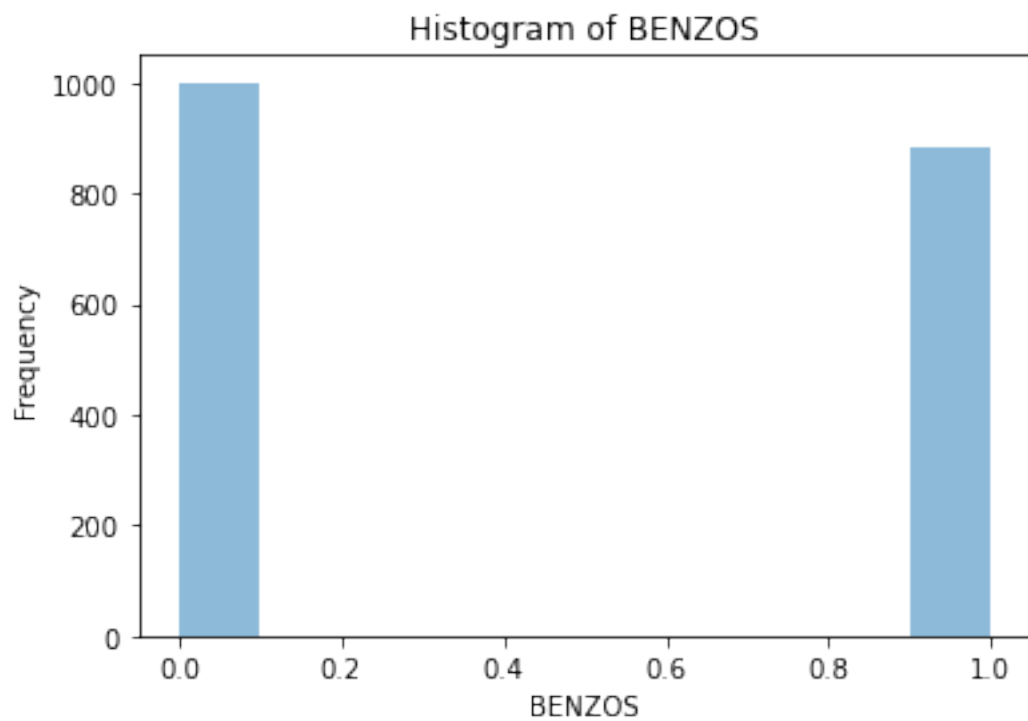
# Load the dataset (User vs Non-user)
data = pd.read_csv('data0.csv')

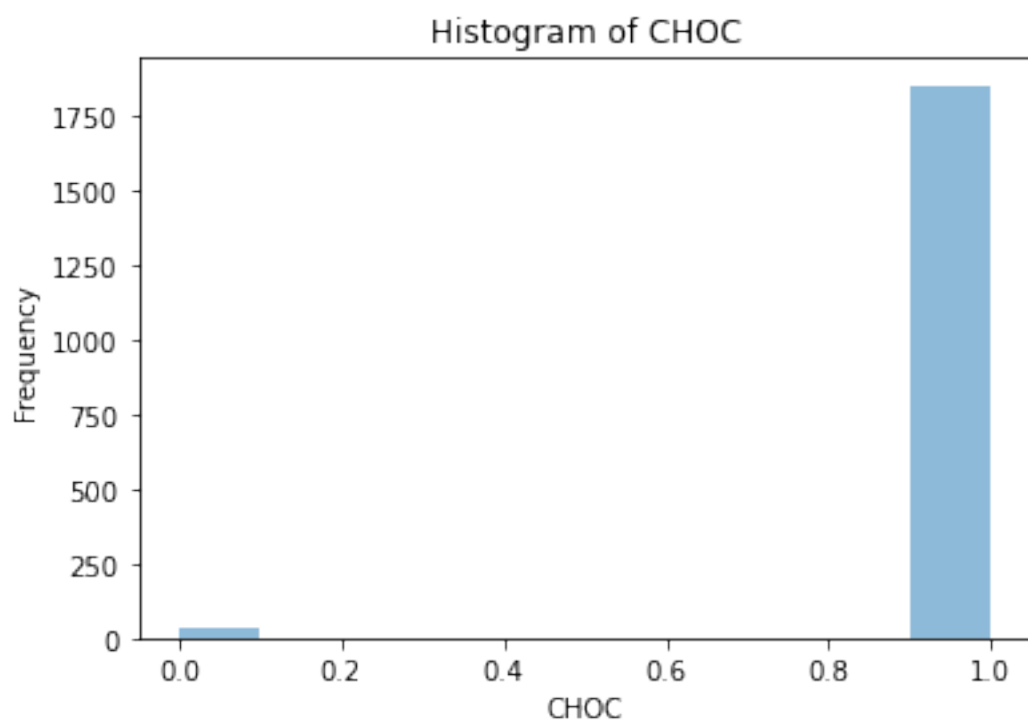
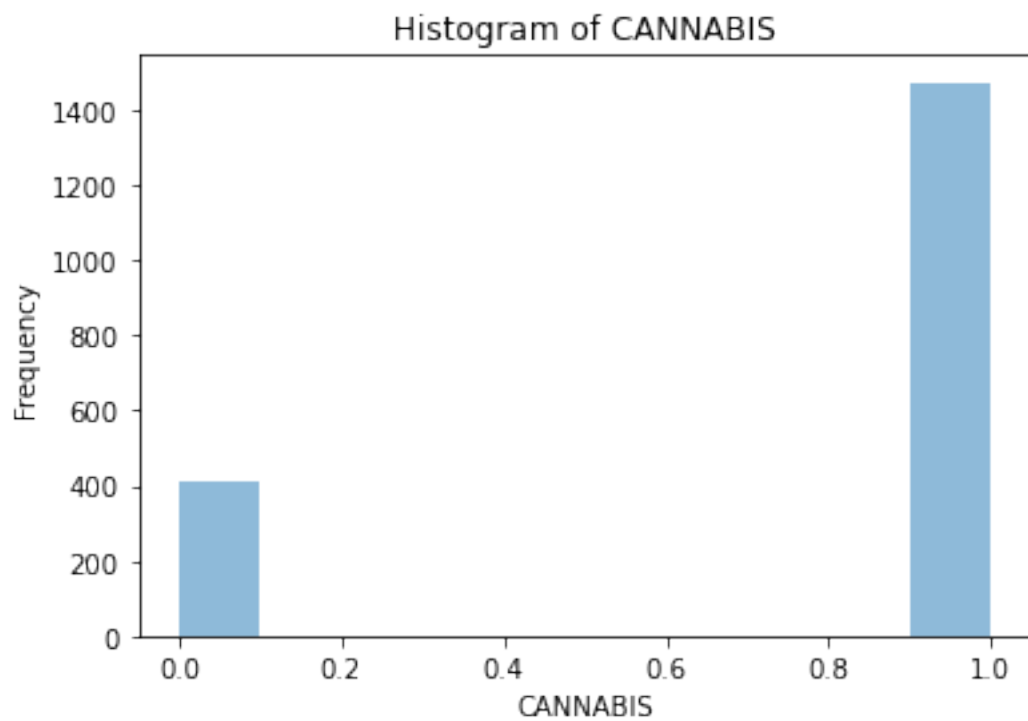
columns_to_plot = data.columns[13:]
```

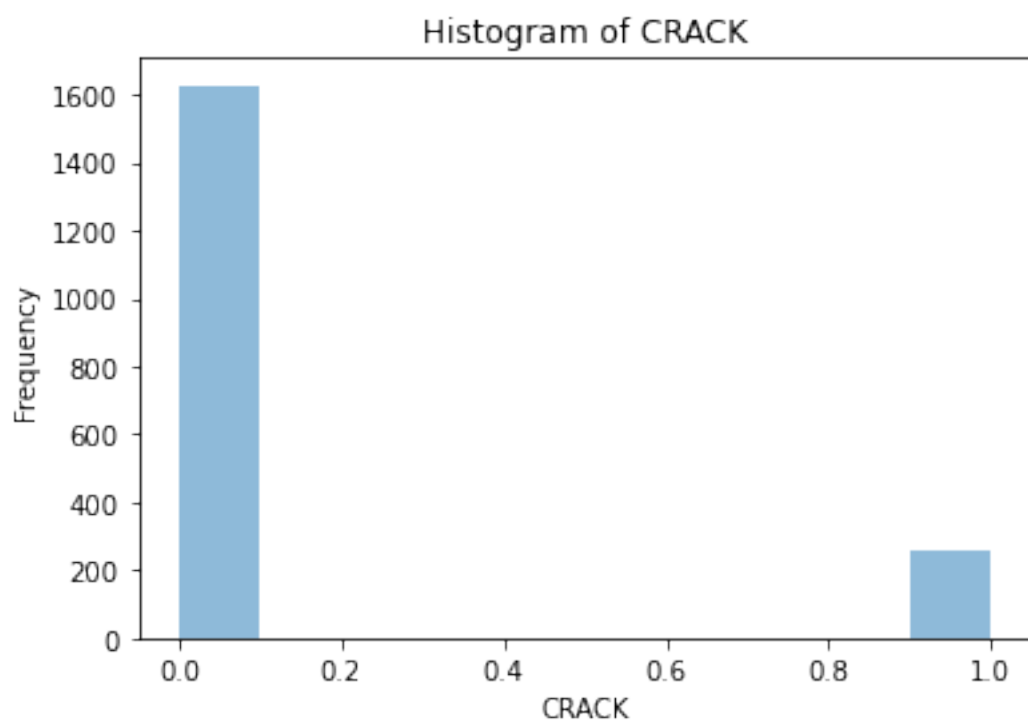
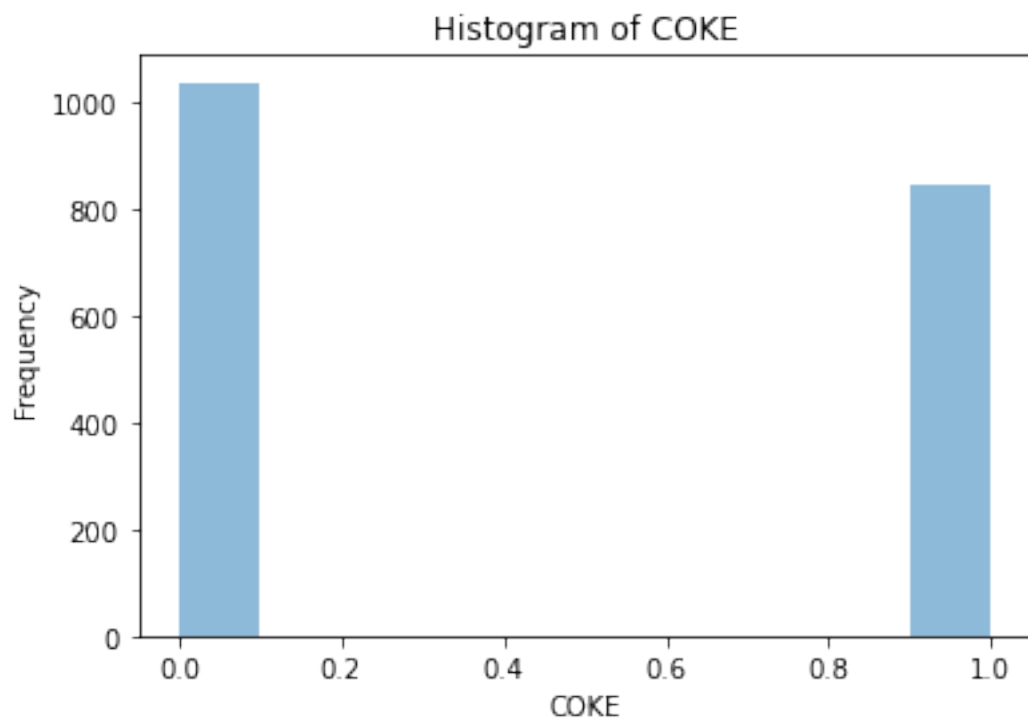
```
# Plot histograms for each column
for col in columns_to_plot:
    plt.hist(data[col], 10, alpha=0.5)
    plt.xlabel(col)
    plt.ylabel('Frequency')
    plt.title('Histogram of {}'.format(col))
    plt.show()
```



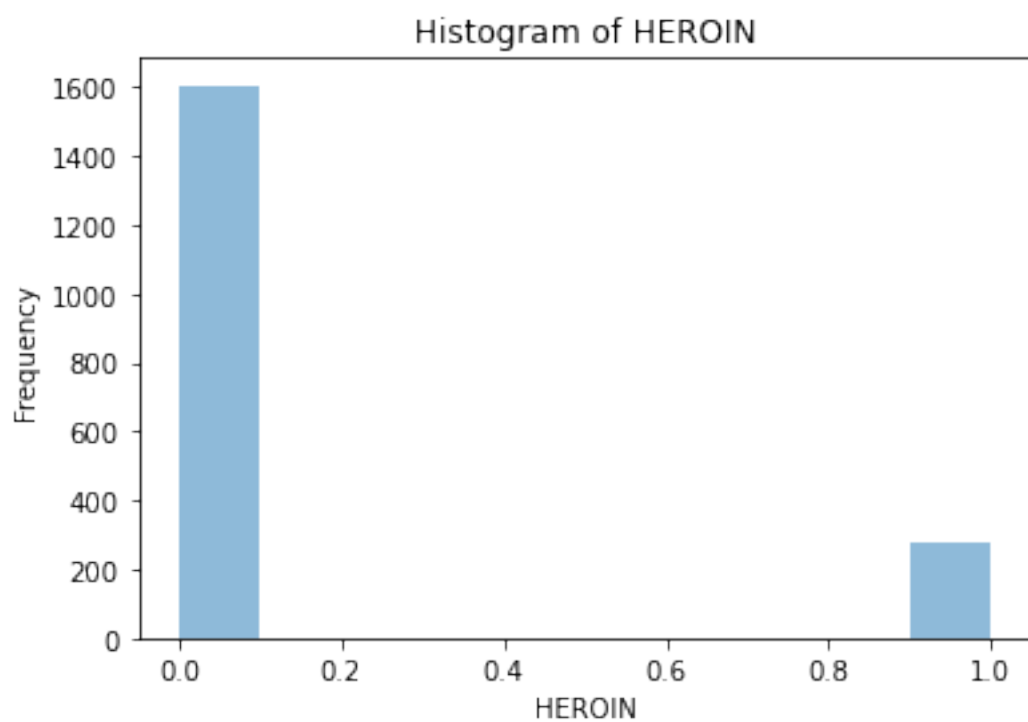
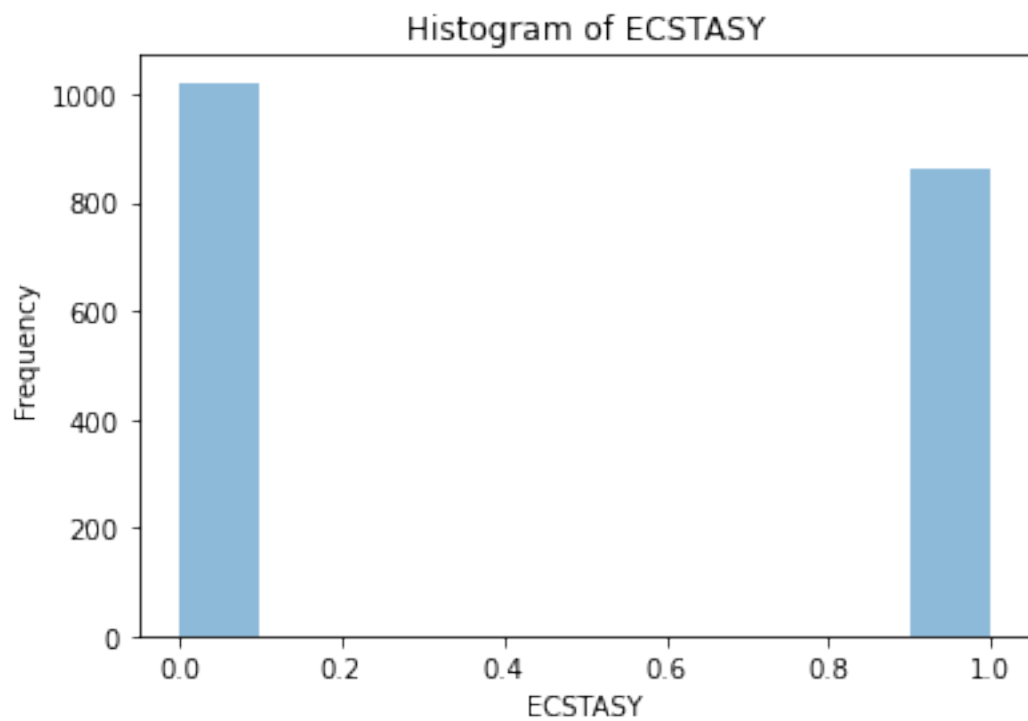


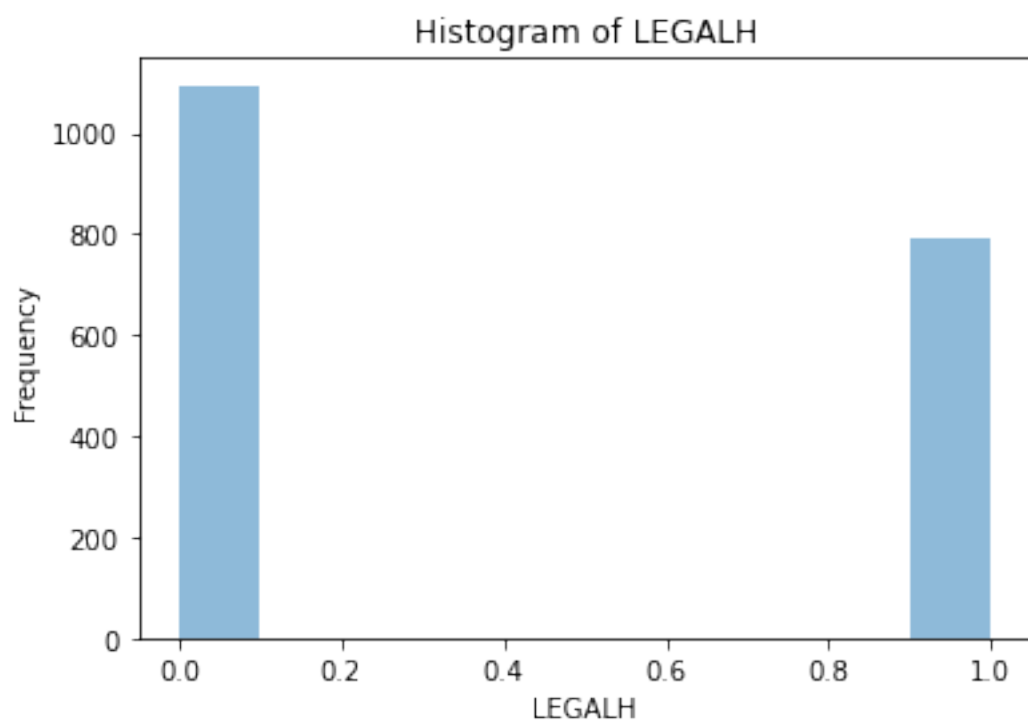
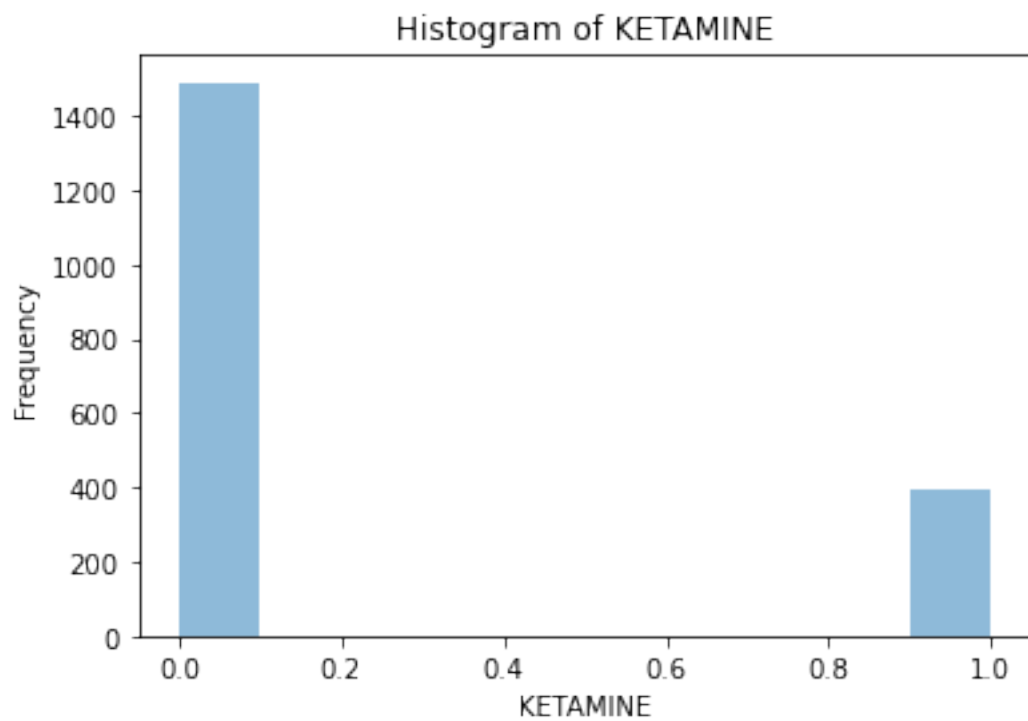


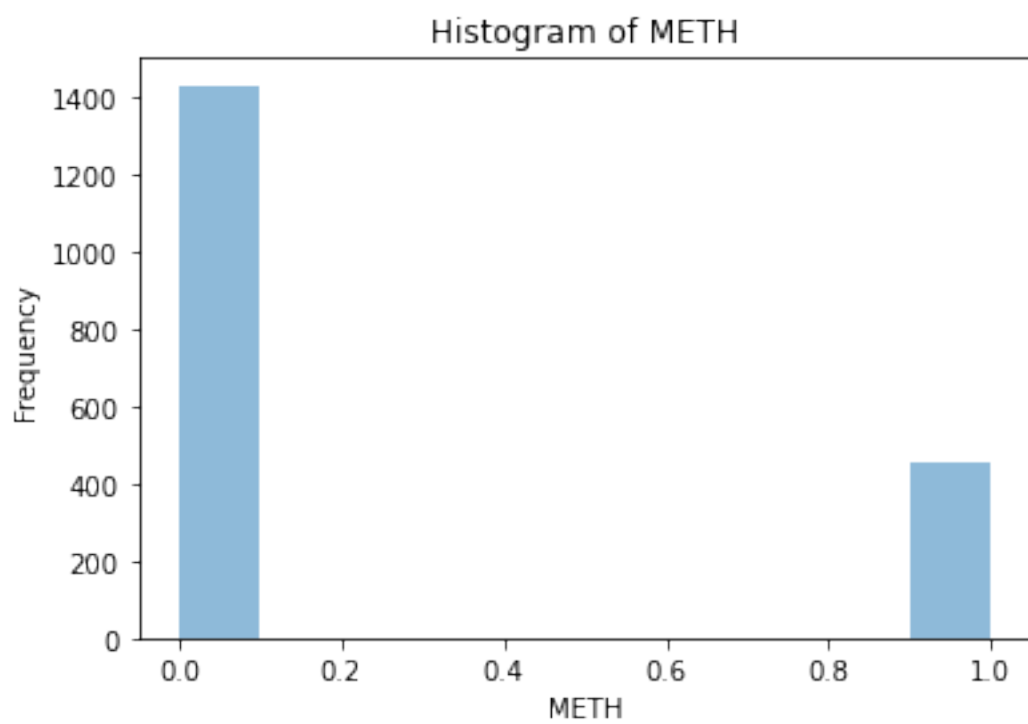
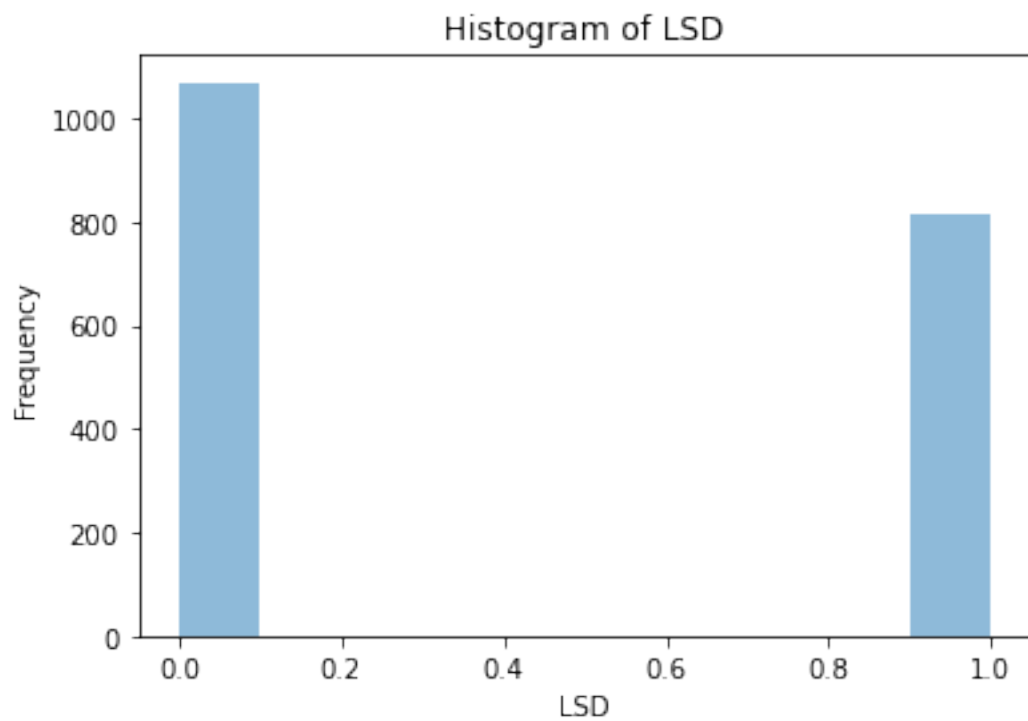


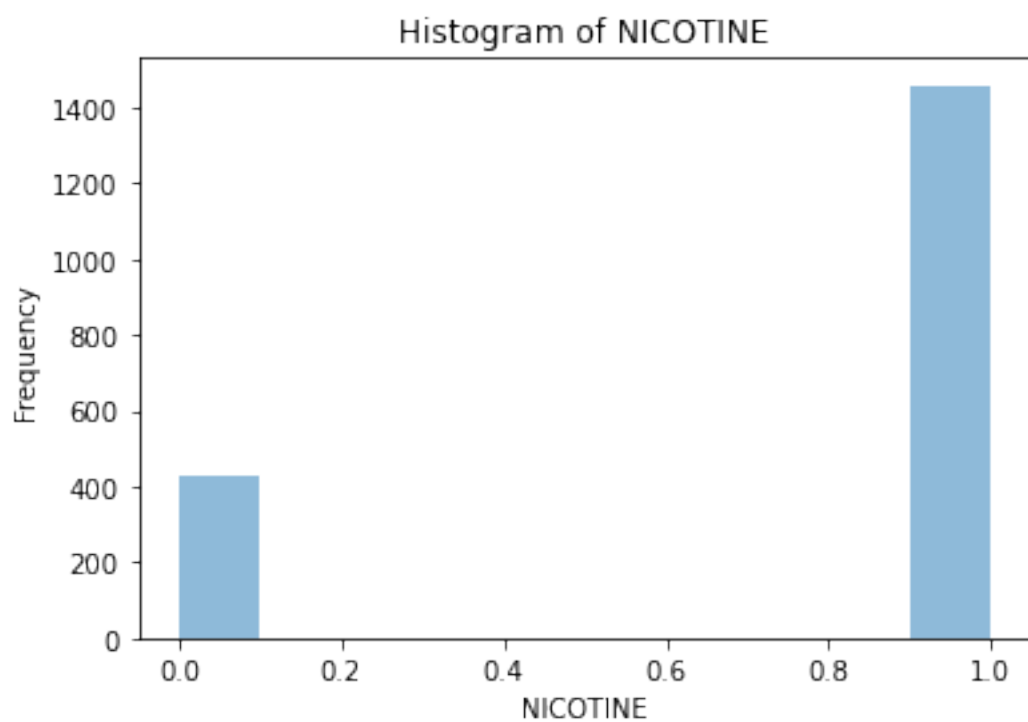
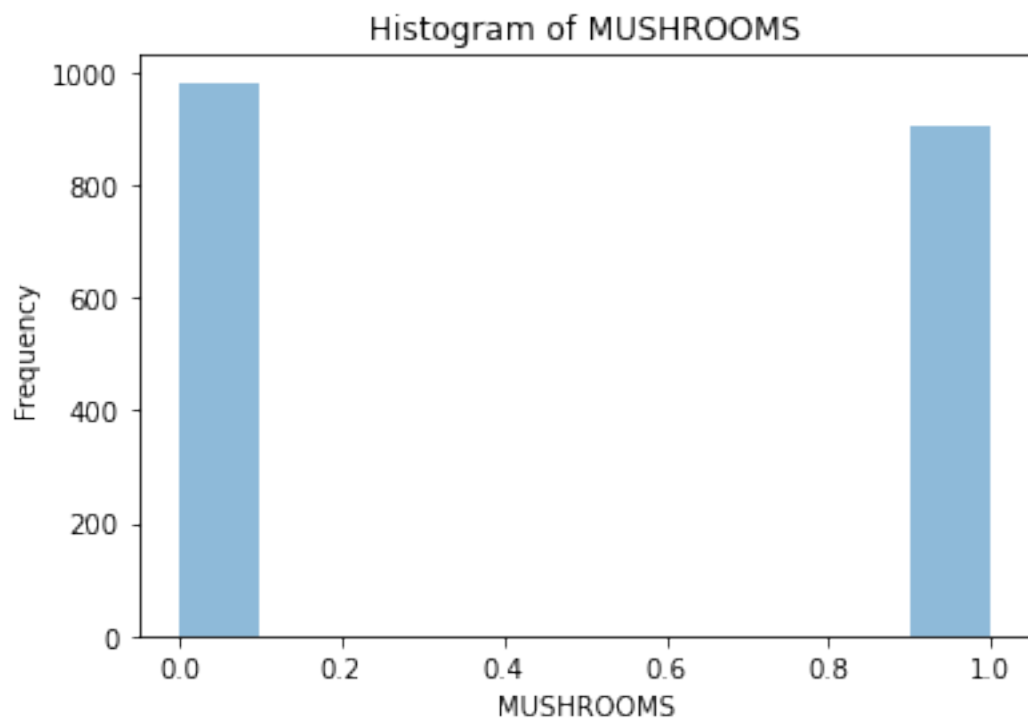


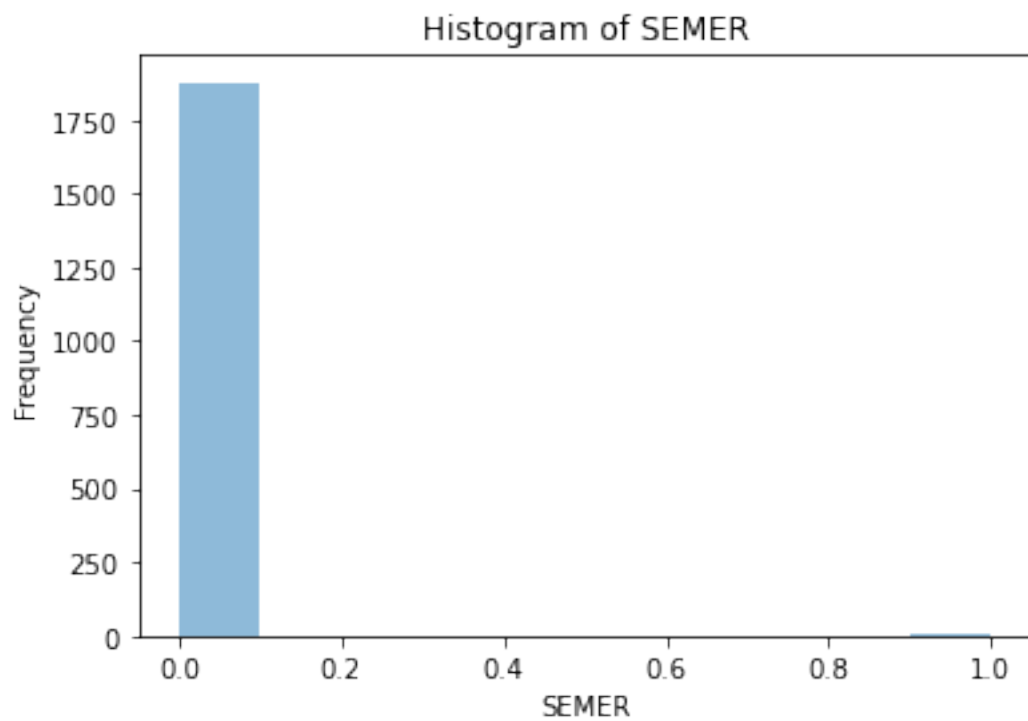








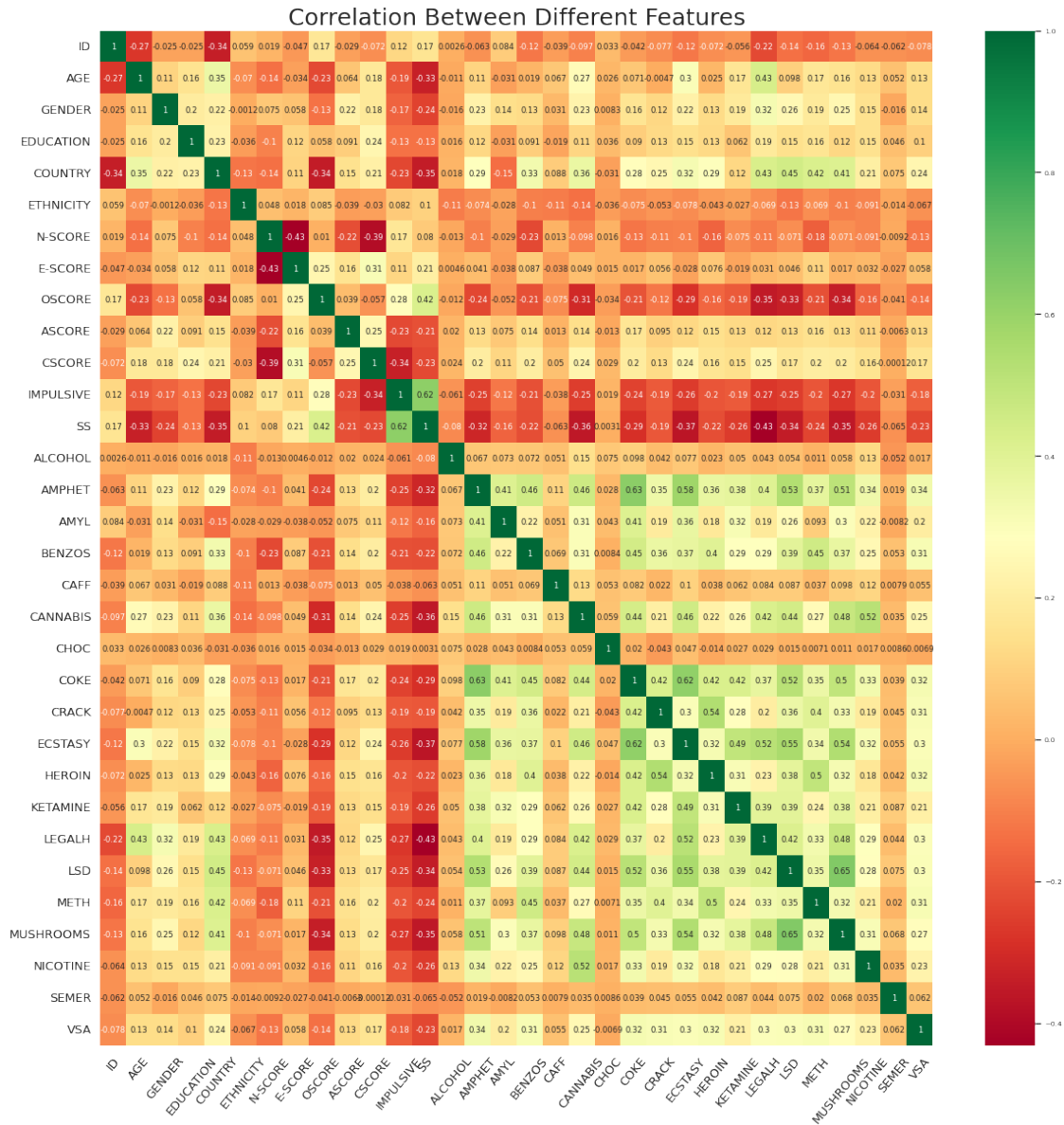




```
[26]: import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
# Read the dataset into a pandas DataFrame
df = pd.read_csv('data0.csv')

# Calculate the correlation matrix
corr = df.corr()

# Plot the correlation matrix using a heatmap
plt.figure(figsize=(20, 20))
sns.set(font_scale=0.7)
hm = sns.heatmap(corr, cmap='RdYlGn', annot=True,
                  yticklabels=df.columns, xticklabels=df.columns)
plt.xticks(fontsize=13, rotation=50)
plt.yticks(fontsize=13)
plt.title("Correlation Between Different Features", fontsize=25)
plt.show()
```



```
[30]: #calculate the correlation matrix
corr_matrix = df.corr()
# Set display options to show all columns
pd.set_option('display.max_columns', None)

print(corr_matrix)
```

	ID	AGE	GENDER	EDUCATION	COUNTRY	ETHNICITY	\
ID	1.000000	-0.271395	-0.025467	-0.025253	-0.340751	0.059309	
AGE	-0.271395	1.000000	0.110286	0.158811	0.354241	-0.069753	
GENDER	-0.025467	0.110286	1.000000	0.196774	0.216271	-0.001213	

EDUCATION	-0.025253	0.158811	0.196774	1.000000	0.225311	-0.036099
COUNTRY	-0.340751	0.354241	0.216271	0.225311	1.000000	-0.127946
ETHNICITY	0.059309	-0.069753	-0.001213	-0.036099	-0.127946	1.000000
N-SCORE	0.018639	-0.136654	0.074646	-0.100993	-0.136191	0.047642
E-SCORE	-0.046960	-0.033849	0.057864	0.115645	0.109524	0.018402
OSCORE	0.173565	-0.226778	-0.131021	0.057994	-0.341969	0.084816
ASCORE	-0.028782	0.063504	0.219743	0.091088	0.150921	-0.038726
CSCORE	-0.072094	0.183564	0.183831	0.240417	0.214000	-0.029923
IMPULSIVE	0.119663	-0.190939	-0.167492	-0.132482	-0.231572	0.082411
SS	0.165882	-0.332188	-0.244277	-0.131146	-0.345415	0.100304
ALCOHOL	0.002603	-0.011043	-0.015873	0.015661	0.018451	-0.107233
AMPHET	-0.063408	0.109436	0.227749	0.121416	0.289936	-0.073625
AMYL	0.084345	-0.031225	0.139876	-0.030558	-0.145600	-0.027764
BENZOS	-0.119580	0.018816	0.128122	0.090525	0.334050	-0.101727
CAFF	-0.038681	0.067046	0.031317	-0.019199	0.087701	-0.109790
CANNABIS	-0.096607	0.266272	0.229856	0.111820	0.357910	-0.135782
CHOC	0.033436	0.025552	0.008283	0.035899	-0.031303	-0.036365
COKE	-0.041876	0.070866	0.164828	0.089617	0.276129	-0.075235
CRACK	-0.077238	-0.004670	0.117092	0.125353	0.246787	-0.053370
ECSTASY	-0.118793	0.303846	0.220971	0.149727	0.322568	-0.078150
HEROIN	-0.071523	0.025361	0.128065	0.127378	0.294199	-0.042506
KETAMINE	-0.055932	0.170027	0.186126	0.062344	0.116845	-0.026879
LEGALH	-0.224968	0.431522	0.320970	0.187153	0.428113	-0.069402
LSD	-0.136590	0.098092	0.256504	0.145237	0.448296	-0.125279
METH	-0.161384	0.174186	0.185520	0.164530	0.419923	-0.069108
MUSHROOMS	-0.127711	0.160399	0.249047	0.120151	0.411946	-0.103812
NICOTINE	-0.064063	0.128249	0.154780	0.151122	0.209948	-0.091127
SEMER	-0.062252	0.051757	-0.016356	0.045503	0.075297	-0.013897
VSA	-0.078460	0.134683	0.136251	0.099730	0.244632	-0.066879

	N-SCORE	E-SCORE	OSCORE	ASCORE	CSCORE	IMPULSIVE \
ID	0.018639	-0.046960	0.173565	-0.028782	-0.072094	0.119663
AGE	-0.136654	-0.033849	-0.226778	0.063504	0.183564	-0.190939
GENDER	0.074646	0.057864	-0.131021	0.219743	0.183831	-0.167492
EDUCATION	-0.100993	0.115645	0.057994	0.091088	0.240417	-0.132482
COUNTRY	-0.136191	0.109524	-0.341969	0.150921	0.214000	-0.231572
ETHNICITY	0.047642	0.018402	0.084816	-0.038726	-0.029923	0.082411
N-SCORE	1.000000	-0.431051	0.010177	-0.216964	-0.391088	0.174399
E-SCORE	-0.431051	1.000000	0.245277	0.157336	0.308024	0.114151
OSCORE	0.010177	0.245277	1.000000	0.038516	-0.056811	0.277512
ASCORE	-0.216964	0.157336	0.038516	1.000000	0.247482	-0.229690
CSCORE	-0.391088	0.308024	-0.056811	0.247482	1.000000	-0.335133
IMPULSIVE	0.174399	0.114151	0.277512	-0.229690	-0.335133	1.000000
SS	0.079988	0.210130	0.421709	-0.208061	-0.229038	0.623120
ALCOHOL	-0.013284	0.004553	-0.011803	0.019570	0.024273	-0.060795
AMPHET	-0.102905	0.041296	-0.235037	0.130153	0.204528	-0.251834
AMYL	-0.029134	-0.038423	-0.051653	0.074661	0.111427	-0.120371
BENZOS	-0.227671	0.087500	-0.211671	0.141013	0.201686	-0.209458



CAFF	0.013012	-0.037894	-0.074668	0.012602	0.050150	-0.037821
CANNABIS	-0.098475	0.048587	-0.307601	0.143676	0.239013	-0.252887
CHOC	0.015906	0.014627	-0.034478	-0.013312	0.029272	0.019225
COKE	-0.129008	0.016512	-0.205020	0.172139	0.197436	-0.242221
CRACK	-0.105372	0.056211	-0.120893	0.094555	0.128098	-0.186298
ECSTASY	-0.101145	-0.027586	-0.285375	0.120317	0.236710	-0.259693
HEROIN	-0.164255	0.076481	-0.157660	0.147990	0.164582	-0.196748
KETAMINE	-0.074501	-0.019191	-0.187070	0.126678	0.148236	-0.189862
LEGALH	-0.109699	0.030952	-0.351021	0.119256	0.245031	-0.271912
LSD	-0.070649	0.045878	-0.331661	0.132080	0.173810	-0.252808
METH	-0.178385	0.108670	-0.208408	0.155313	0.204428	-0.199169
MUSHROOMS	-0.071024	0.016850	-0.340729	0.134405	0.197865	-0.270494
NICOTINE	-0.091075	0.032325	-0.164933	0.111990	0.156644	-0.199119
SEMER	-0.009179	-0.027472	-0.041361	-0.006335	-0.000122	-0.030864
VSA	-0.125435	0.058456	-0.136549	0.125917	0.165344	-0.176152

	SS	ALCOHOL	AMPHET	AMYL	BENZOS	CAFF \
ID	0.165882	0.002603	-0.063408	0.084345	-0.119580	-0.038681
AGE	-0.332188	-0.011043	0.109436	-0.031225	0.018816	0.067046
GENDER	-0.244277	-0.015873	0.227749	0.139876	0.128122	0.031317
EDUCATION	-0.131146	0.015661	0.121416	-0.030558	0.090525	-0.019199
COUNTRY	-0.345415	0.018451	0.289936	-0.145600	0.334050	0.087701
ETHNICITY	0.100304	-0.107233	-0.073625	-0.027764	-0.101727	-0.109790
N-SCORE	0.079988	-0.013284	-0.102905	-0.029134	-0.227671	0.013012
E-SCORE	0.210130	0.004553	0.041296	-0.038423	0.087500	-0.037894
OSCORE	0.421709	-0.011803	-0.235037	-0.051653	-0.211671	-0.074668
ASCORE	-0.208061	0.019570	0.130153	0.074661	0.141013	0.012602
CSCORE	-0.229038	0.024273	0.204528	0.111427	0.201686	0.050150
IMPULSIVE	0.623120	-0.060795	-0.251834	-0.120371	-0.209458	-0.037821
SS	1.000000	-0.080036	-0.317712	-0.163348	-0.215194	-0.062942
ALCOHOL	-0.080036	1.000000	0.066976	0.073080	0.071589	0.050758
AMPHET	-0.317712	0.066976	1.000000	0.410162	0.455732	0.107401
AMYL	-0.163348	0.073080	0.410162	1.000000	0.224999	0.051344
BENZOS	-0.215194	0.071589	0.455732	0.224999	1.000000	0.068674
CAFF	-0.062942	0.050758	0.107401	0.051344	0.068674	1.000000
CANNABIS	-0.357194	0.149861	0.462418	0.305886	0.305562	0.130437
CHOC	0.003091	0.074761	0.028200	0.043120	0.008425	0.053282
COKE	-0.290724	0.098387	0.626542	0.409890	0.445211	0.081967
CRACK	-0.191119	0.042372	0.347762	0.192676	0.358350	0.022024
ECSTASY	-0.368999	0.076674	0.584585	0.364840	0.371968	0.101932
HEROIN	-0.215591	0.022982	0.364135	0.183729	0.399152	0.037796
KETAMINE	-0.261913	0.050193	0.376999	0.317618	0.288737	0.062068
LEGALH	-0.426431	0.042547	0.403499	0.187764	0.285680	0.084409
LSD	-0.343731	0.054050	0.534619	0.264283	0.390227	0.087299
METH	-0.236490	0.011402	0.374612	0.093117	0.454023	0.036819
MUSHROOMS	-0.353802	0.058150	0.506968	0.299467	0.368213	0.097723
NICOTINE	-0.259361	0.126363	0.340590	0.224141	0.251061	0.115842
SEMER	-0.065438	-0.052470	0.018653	-0.008161	0.053045	0.007870

VSA	-0.234698	0.016681	0.335598	0.204598	0.314416	0.054893
-----	-----------	----------	----------	----------	----------	----------

	CANNABIS	CHOC	COKE	CRACK	ECSTASY	HEROIN \
ID	-0.096607	0.033436	-0.041876	-0.077238	-0.118793	-0.071523
AGE	0.266272	0.025552	0.070866	-0.004670	0.303846	0.025361
GENDER	0.229856	0.008283	0.164828	0.117092	0.220971	0.128065
EDUCATION	0.111820	0.035899	0.089617	0.125353	0.149727	0.127378
COUNTRY	0.357910	-0.031303	0.276129	0.246787	0.322568	0.294199
ETHNICITY	-0.135782	-0.036365	-0.075235	-0.053370	-0.078150	-0.042506
N-SCORE	-0.098475	0.015906	-0.129008	-0.105372	-0.101145	-0.164255
E-SCORE	0.048587	0.014627	0.016512	0.056211	-0.027586	0.076481
OSCORE	-0.307601	-0.034478	-0.205020	-0.120893	-0.285375	-0.157660
ASCORE	0.143676	-0.013312	0.172139	0.094555	0.120317	0.147990
CSCORE	0.239013	0.029272	0.197436	0.128098	0.236710	0.164582
IMPULSIVE	-0.252887	0.019225	-0.242221	-0.186298	-0.259693	-0.196748
SS	-0.357194	0.003091	-0.290724	-0.191119	-0.368999	-0.215591
ALCOHOL	0.149861	0.074761	0.098387	0.042372	0.076674	0.022982
AMPHET	0.462418	0.028200	0.626542	0.347762	0.584585	0.364135
AMYL	0.305886	0.043120	0.409890	0.192676	0.364840	0.183729
BENZOS	0.305562	0.008425	0.445211	0.358350	0.371968	0.399152
CAFF	0.130437	0.053282	0.081967	0.022024	0.101932	0.037796
CANNABIS	1.000000	0.059458	0.437227	0.210929	0.456377	0.217633
CHOC	0.059458	1.000000	0.019639	-0.043254	0.046710	-0.014396
COKE	0.437227	0.019639	1.000000	0.419112	0.618099	0.423394
CRACK	0.210929	-0.043254	0.419112	1.000000	0.302782	0.541097
ECSTASY	0.456377	0.046710	0.618099	0.302782	1.000000	0.319322
HEROIN	0.217633	-0.014396	0.423394	0.541097	0.319322	1.000000
KETAMINE	0.256969	0.027300	0.420617	0.284190	0.494306	0.309076
LEGALH	0.419216	0.028527	0.370789	0.202468	0.518744	0.231250
LSD	0.444663	0.015354	0.517327	0.356101	0.554367	0.381722
METH	0.266274	0.007107	0.353894	0.398579	0.335617	0.495559
MUSHROOMS	0.477128	0.010929	0.495821	0.325668	0.539440	0.322096
NICOTINE	0.518081	0.017000	0.331755	0.190038	0.315594	0.176527
SEMER	0.034581	0.008579	0.039461	0.045231	0.054591	0.041572
VSA	0.254334	-0.006853	0.319694	0.309510	0.304243	0.316840

	KETAMINE	LEGALH	LSD	METH	MUSHROOMS	NICOTINE \
ID	-0.055932	-0.224968	-0.136590	-0.161384	-0.127711	-0.064063
AGE	0.170027	0.431522	0.098092	0.174186	0.160399	0.128249
GENDER	0.186126	0.320970	0.256504	0.185520	0.249047	0.154780
EDUCATION	0.062344	0.187153	0.145237	0.164530	0.120151	0.151122
COUNTRY	0.116845	0.428113	0.448296	0.419923	0.411946	0.209948
ETHNICITY	-0.026879	-0.069402	-0.125279	-0.069108	-0.103812	-0.091127
N-SCORE	-0.074501	-0.109699	-0.070649	-0.178385	-0.071024	-0.091075
E-SCORE	-0.019191	0.030952	0.045878	0.108670	0.016850	0.032325
OSCORE	-0.187070	-0.351021	-0.331661	-0.208408	-0.340729	-0.164933
ASCORE	0.126678	0.119256	0.132080	0.155313	0.134405	0.111990
CSCORE	0.148236	0.245031	0.173810	0.204428	0.197865	0.156644

IMPULSIVE	-0.189862	-0.271912	-0.252808	-0.199169	-0.270494	-0.199119
SS	-0.261913	-0.426431	-0.343731	-0.236490	-0.353802	-0.259361
ALCOHOL	0.050193	0.042547	0.054050	0.011402	0.058150	0.126363
AMPHET	0.376999	0.403499	0.534619	0.374612	0.506968	0.340590
AMYL	0.317618	0.187764	0.264283	0.093117	0.299467	0.224141
BENZOS	0.288737	0.285680	0.390227	0.454023	0.368213	0.251061
CAFF	0.062068	0.084409	0.087299	0.036819	0.097723	0.115842
CANNABIS	0.256969	0.419216	0.444663	0.266274	0.477128	0.518081
CHOC	0.027300	0.028527	0.015354	0.007107	0.010929	0.017000
COKE	0.420617	0.370789	0.517327	0.353894	0.495821	0.331755
CRACK	0.284190	0.202468	0.356101	0.398579	0.325668	0.190038
ECSTASY	0.494306	0.518744	0.554367	0.335617	0.539440	0.315594
HEROIN	0.309076	0.231250	0.381722	0.495559	0.322096	0.176527
KETAMINE	1.000000	0.386286	0.389377	0.244863	0.375154	0.210607
LEGALH	0.386286	1.000000	0.415659	0.330472	0.480028	0.286372
LSD	0.389377	0.415659	1.000000	0.349035	0.649625	0.279291
METH	0.244863	0.330472	0.349035	1.000000	0.323749	0.208584
MUSHROOMS	0.375154	0.480028	0.649625	0.323749	1.000000	0.311871
NICOTINE	0.210607	0.286372	0.279291	0.208584	0.311871	1.000000
SEMER	0.086694	0.043705	0.074723	0.020290	0.068081	0.035384
VSA	0.214010	0.296043	0.303274	0.312872	0.273346	0.231268

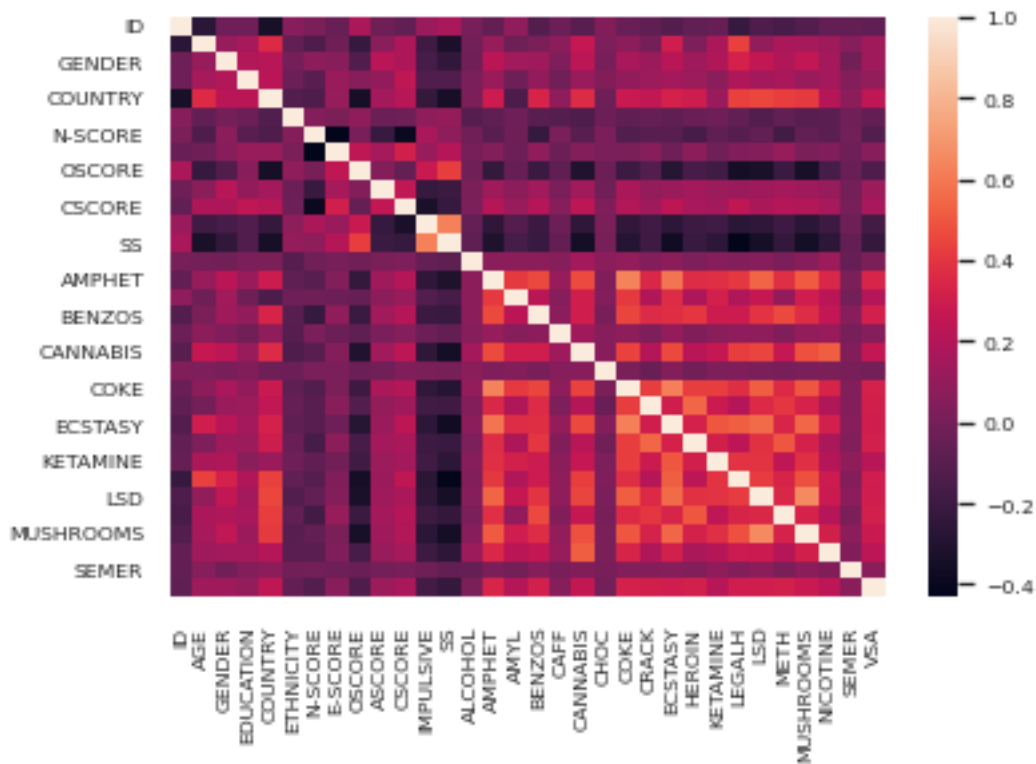
	SEMER	VSA
ID	-0.062252	-0.078460
AGE	0.051757	0.134683
GENDER	-0.016356	0.136251
EDUCATION	0.045503	0.099730
COUNTRY	0.075297	0.244632
ETHNICITY	-0.013897	-0.066879
N-SCORE	-0.009179	-0.125435
E-SCORE	-0.027472	0.058456
OSCORE	-0.041361	-0.136549
ASCORE	-0.006335	0.125917
CSCORE	-0.000122	0.165344
IMPULSIVE	-0.030864	-0.176152
SS	-0.065438	-0.234698
ALCOHOL	-0.052470	0.016681
AMPHET	0.018653	0.335598
AMYL	-0.008161	0.204598
BENZOS	0.053045	0.314416
CAFF	0.007870	0.054893
CANNABIS	0.034581	0.254334
CHOC	0.008579	-0.006853
COKE	0.039461	0.319694
CRACK	0.045231	0.309510
ECSTASY	0.054591	0.304243
HEROIN	0.041572	0.316840
KETAMINE	0.086694	0.214010

LEGALH	0.043705	0.296043
LSD	0.074723	0.303274
METH	0.020290	0.312872
MUSHROOMS	0.068081	0.273346
NICOTINE	0.035384	0.231268
SEMER	1.000000	0.061748
VSA	0.061748	1.000000

Correlation Heat Map

```
[37]: sns.heatmap(df.corr())
```

```
[37]: <AxesSubplot:>
```



## 10.2 Finalizing Data Attributes Based On Correlation Analysis

```
[43]: df = pd.read_csv('data0.csv')

# Assuming 'df' is your DataFrame
# Select columns with numerical data only
```

```

# Calculate the correlation matrix
# Computes the correlation matrix using the corr() function
# This matrix contains correlation coefficients between all pairs of
↪numerical columns.
numerical_columns = df.select_dtypes(include=['number'])
corr_matrix = numerical_columns.corr()

# Set a threshold for correlation
correlation_threshold = 0.35 # Adjust this threshold as needed

# Find columns highly correlated with each other
# Iterates through the correlation matrix to identify pairs of columns that
↪have a correlation greater than threshold
highly_correlated_cols = set()
# Finding highly correlated columns
highly_correlated_cols = set()
for i in range(len(corr_matrix.columns)):
    for j in range(i):
        if abs(corr_matrix.iloc[i, j]) > correlation_threshold:
            colname_i = corr_matrix.columns[i]
            colname_j = corr_matrix.columns[j]
            highly_correlated_cols.add(colname_i)
            highly_correlated_cols.add(colname_j)

# Selecting unique columns based on the correlation threshold

selected_columns = list(set(df.columns) - highly_correlated_cols)
highly_correlated_cols = list(highly_correlated_cols)

# Displaying selected columns
print("High Correlation Columns:")
print(highly_correlated_cols)
print()
print("Low Correlation Columns")
print(selected_columns)

```

High Correlation Columns:

```
['CRACK', 'LEGALH', 'AMYL', 'CSCORE', 'N-SCORE', 'AGE', 'HEROIN', 'IMPULSIVE',
'AMPHET', 'SS', 'OSCORE', 'E-SCORE', 'ECSTASY', 'COUNTRY', 'KETAMINE', 'LSD',
'MUSHROOMS', 'NICOTINE', 'BENZOS', 'COKE', 'METH', 'CANNABIS']
```

Low Correlation Columns

```
['ALCOHOL', 'SEMER', 'VSA', 'ID', 'CHOC', 'CAFF', 'GENDER', 'ETHNICITY',
'EDUCATION', 'ASCORE']
```

### 10.3 Findings

The code systematically compares each pair of columns in the correlation matrix and identifies those that surpass the specified correlation threshold. Based on the correlation analysis, using a correlation threshold of 0.35, we will use the following data attributes for each respondent from the dataset in order to train machine learning algorithms. - AGE - COUNTRY - N-SCORE - E-SCORE - OSCORE - IMPULSIVE - SS - AMPHET - AMYL - BENZOS - CANNABIS - COKE - CRACK - ECSTASY - HEROIN - KETAMINE - LEGALH - LSD - METH - MUSHROOMS - NICOTINE

The main takeaway is that the following data attributes cannot be used in this project due to their weaker relationships or lack of dependencies with other columns in the dataset. To prevent negatively affecting the performance of the algorithm and models, the following data attributes will not be used and have been removed from the data. - GENDER - ETHNICITY - EDUCATION - ASCORE - ALCOHOL - CAFF - CHOC - SEMER - VSA

## 11 Feature Selection

For the project, we are selecting the following as Features (X) to train the programs, as they are a comprehensive set of features that can be easily identified and determined in individuals. The following are relevant and impactful features that come from the results of the correlation analysis (X). - AGE - COUNTRY - N-SCORE - E-SCORE - OSCORE - CSCORE - IMPULSIVE - SS

The following are the target variables (Y) for the project. - AMPHET - AMYL - BENZOS - CANNABIS - COKE - CRACK - ECSTASY - HEROIN - KETAMINE - LEGALH - LSD - METH - MUSHROOMS - NICOTINE

## 12 Model Training and Learning

We used a Supervised Learning setting, where we trained the models using features (X) to predict the target variable (Y) to maximize model performance. This ensured the model learned the relationship between input features and output variable based on label data. The models learn patterns from the training data and then predicts on the unseen testing data to evaluate how accurate it is to new, unseen data. We used a 70% training to 30% testing ratio for the project while using train-test split. After, we evaluated the effectiveness of the model's predictions using cross-validation with 5 folds. We used the results from the cross-validation analysis to evaluate the effectiveness of the model for a specific drug because it reduces variability in model performance estimation as train-test splits can produce variability due to the randomness in the data-split. Cross-validation ensured the results are more comprehensive by using different subsets of the data. We determined the AUC score and the ROC graph using cross-validation. We used the following machine learning models in this study, and evaluated the effectiveness and accuracy of each for all 14 drugs. - Linear Discriminant Analysis (LDA) - K-Nearest Neighbors (KNN) - Decision Tree (CART) - Gaussian Naive Bayes (NB) - Random Forest (RF) - Gradient Boosting Machine (GBM) - Support Vector Machine (SVM) - Logistic Regression (LR)

## 12.1 Linear Discriminant Analysis (LDA)

```
[3]: from sklearn.metrics import roc_auc_score
      from sklearn.neighbors import KNeighborsClassifier
      import pandas as pd
      from sklearn.model_selection import train_test_split, cross_val_score, \
      ↪StratifiedKFold
      from sklearn.ensemble import RandomForestClassifier, \
      ↪GradientBoostingClassifier, ExtraTreesClassifier
      from sklearn.metrics import accuracy_score, classification_report, \
      ↪confusion_matrix
      from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
      from sklearn.tree import DecisionTreeClassifier
      from sklearn.naive_bayes import GaussianNB
      from sklearn.svm import SVC, LinearSVC, NuSVC
      from sklearn.neighbors import KNeighborsClassifier, \
      ↪RadiusNeighborsClassifier, NearestCentroid
      from sklearn.neural_network import MLPClassifier
      from pandas import read_csv
      from pandas.plotting import scatter_matrix
      from matplotlib import pyplot as plt
      from scipy.stats import pearsonr
      import seaborn as sns
      from sklearn.model_selection import train_test_split
      from sklearn.linear_model import LinearRegression
      from sklearn.metrics import accuracy_score, precision_score, recall_score, \
      ↪f1_score, confusion_matrix, roc_curve
      import matplotlib.pyplot as plt
      import numpy as np
      from sklearn.linear_model import LogisticRegression
      from sklearn.metrics import roc_auc_score
      from sklearn.metrics import roc_curve
      from sklearn.metrics import accuracy_score
      from sklearn.model_selection import train_test_split
      from sklearn.metrics import accuracy_score, precision_score, recall_score, \
      ↪f1_score, confusion_matrix
      from sklearn.model_selection import cross_val_score
      from sklearn.model_selection import cross_val_predict
```

```
[4]: df = pd.read_csv('data6.csv')
      print("")
      print("Model: Linear Discriminant Analysis")

      for col_index in range(9, len(df.columns)):
```

```

columnName = df.columns[col_index]
print()
print("--" + (columnName))
print()

print('Train-Test Split:')

X = df.iloc[:, 1:9]
y = df[columnName]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
↪random_state=42)

# Create an instance of the model
model_1 = LinearDiscriminantAnalysis()

# Train the model on the training data
model_1.fit(X_train, y_train)

# Make predictions on the test data
y_pred1 = model_1.predict(X_test)

# Calculate evaluation metrics
accuracy = accuracy_score(y_test, y_pred1)
precision = precision_score(y_test, y_pred1)
recall = recall_score(y_test, y_pred1)
f1 = f1_score(y_test, y_pred1)
conf_matrix = confusion_matrix(y_test, y_pred1)

# Print the evaluation metrics
print(f"Accuracy: {accuracy:.3f}")
print(f"Precision: {precision:.3f}")
print(f"Recall: {recall:.3f}")
print(f"F1 score: {f1:.3f}")
print("Confusion matrix:")
print(conf_matrix)
print()

N_FOLDS = 5
# Perform cross-validation predictions
y_scores = cross_val_predict(model_1, X_train, y_train, cv=N_FOLDS,
↪method="predict_proba")

```



```

fpr, tpr, thresholds = roc_curve(y_train, y_scores[:, 1])
roc_auc = roc_auc_score(y_train, y_scores[:, 1])

print('Cross Validation:')
cv_accuracy = cross_val_score(model_1, X, y, cv=5, scoring='accuracy')
print('Mean Accuracy: %.5f +/- %.5f' % (np.mean(cv_accuracy), np.
↳std(cv_accuracy)))
precision_scores = cross_val_score(model_1, X_train, y_train, cv=N_FOLDS,
↳scoring="precision")
print('Mean Precision: %.5f +/- %.5f' % (np.mean(precision_scores), np.
↳std(precision_scores)))
recall_scores = cross_val_score(model_1, X_train, y_train, cv=N_FOLDS,
↳scoring="recall")
cv_f1 = cross_val_score(model_1, X, y, cv=5, scoring='f1')
print('Mean Recall: %.5f +/- %.5f' % (np.mean(recall_scores), np.
↳std(recall_scores)))
print(f"Mean F1 score: {np.mean(cv_f1):.2f} +/- {np.std(cv_f1):.2f}")
auc_scores = cross_val_score(model_1, X_train, y_train, cv=N_FOLDS,
↳scoring="roc_auc")
print('Mean AUC Score: %.5f +/- %.5f' % (np.mean(auc_scores), np.
↳std(auc_scores)))
print()
print()

# Plot ROC curve
plt.figure()
plt.plot(fpr, tpr, label='ROC curve (area = %0.3f)' % roc_auc)
plt.plot([0, 1], [0, 1], 'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve: LDA - ' + (columnName))
plt.legend(loc="lower right")
plt.show()

```

Model: Linear Discriminant Analysis

--AMPHET

Train-Test Split:

Accuracy: 0.674

Precision: 0.673

Recall: 0.624

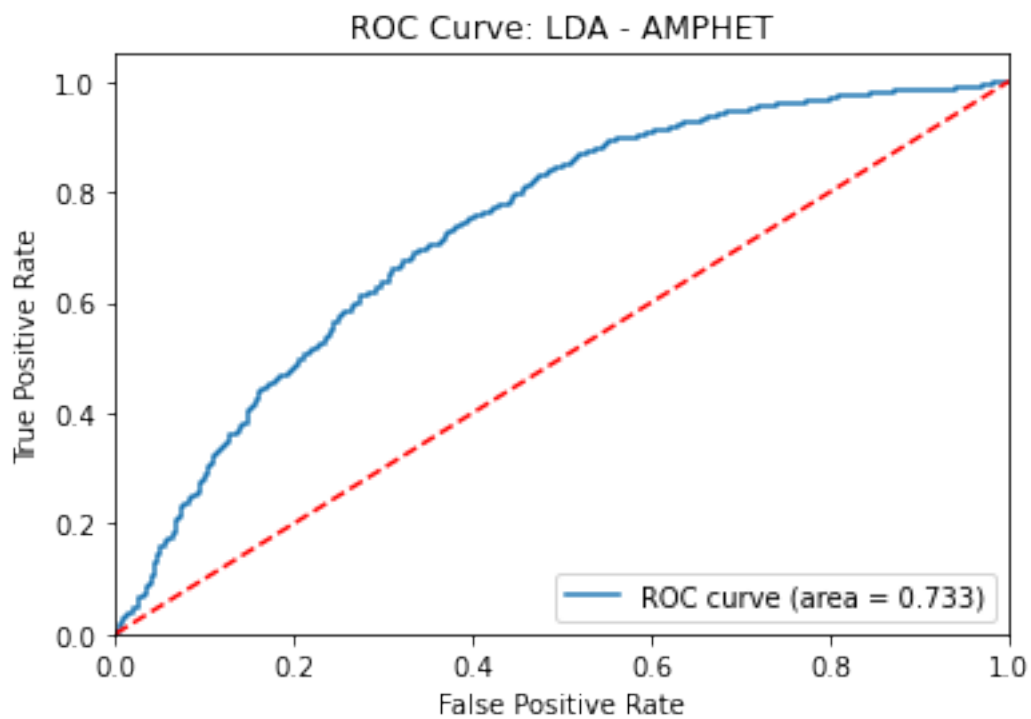
F1 score: 0.648

Confusion matrix:

```
[[141  55]
 [ 68 113]]
```

Cross Validation:

Mean Accuracy: 0.66260 +/- 0.04242  
Mean Precision: 0.66475 +/- 0.01400  
Mean Recall: 0.62774 +/- 0.04424  
Mean F1 score: 0.62 +/- 0.12  
Mean AUC Score: 0.73371 +/- 0.01191



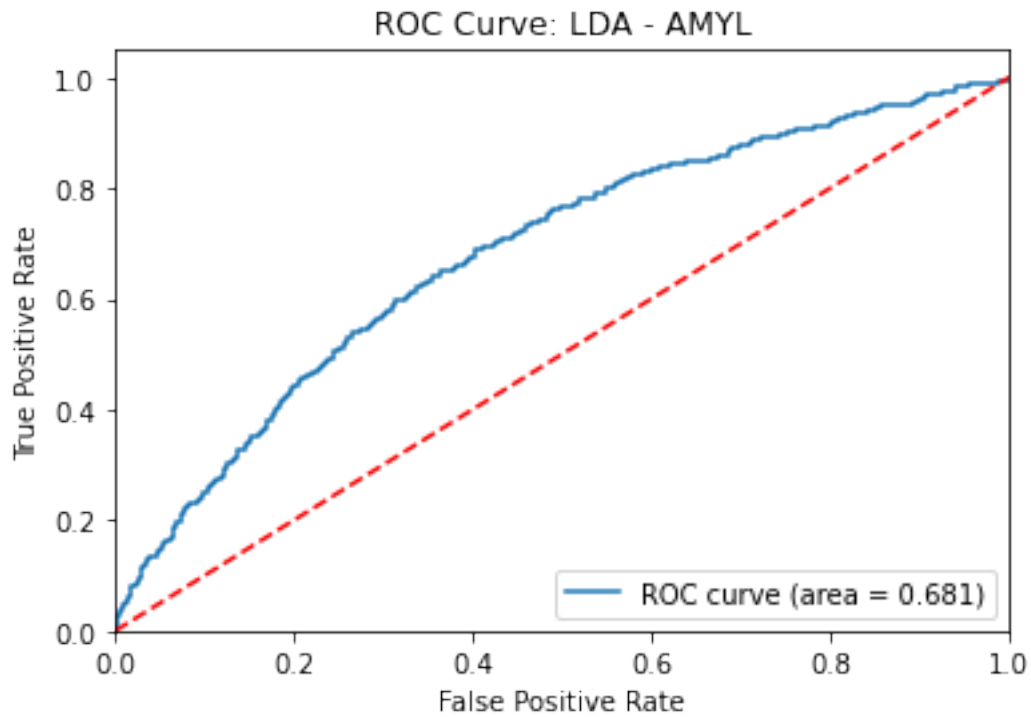
--AMYL

Train-Test Split:

Accuracy: 0.729  
Precision: 0.667  
Recall: 0.241  
F1 score: 0.354  
Confusion matrix:  
[[247 14]
 [ 88 28]]

Cross Validation:

Mean Accuracy: 0.70822 +/- 0.00731  
Mean Precision: 0.55157 +/- 0.06055  
Mean Recall: 0.18102 +/- 0.01834  
Mean F1 score: 0.32 +/- 0.06  
Mean AUC Score: 0.68375 +/- 0.04221



--BENZOS

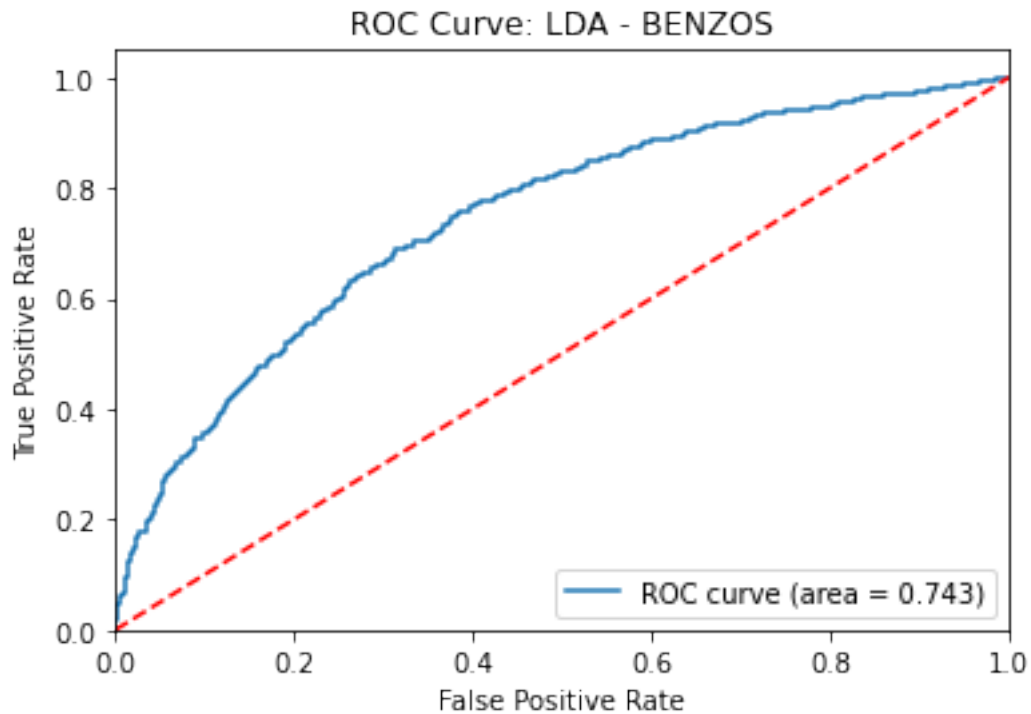
Train-Test Split:

Accuracy: 0.700  
Precision: 0.694  
Recall: 0.626  
F1 score: 0.659  
Confusion matrix:  
[[155 48]  
 [ 65 109]]

Cross Validation:

Mean Accuracy: 0.68223 +/- 0.04069

Mean Precision: 0.68062 +/- 0.04106  
Mean Recall: 0.62576 +/- 0.05435  
Mean F1 score: 0.63 +/- 0.12  
Mean AUC Score: 0.74506 +/- 0.03029

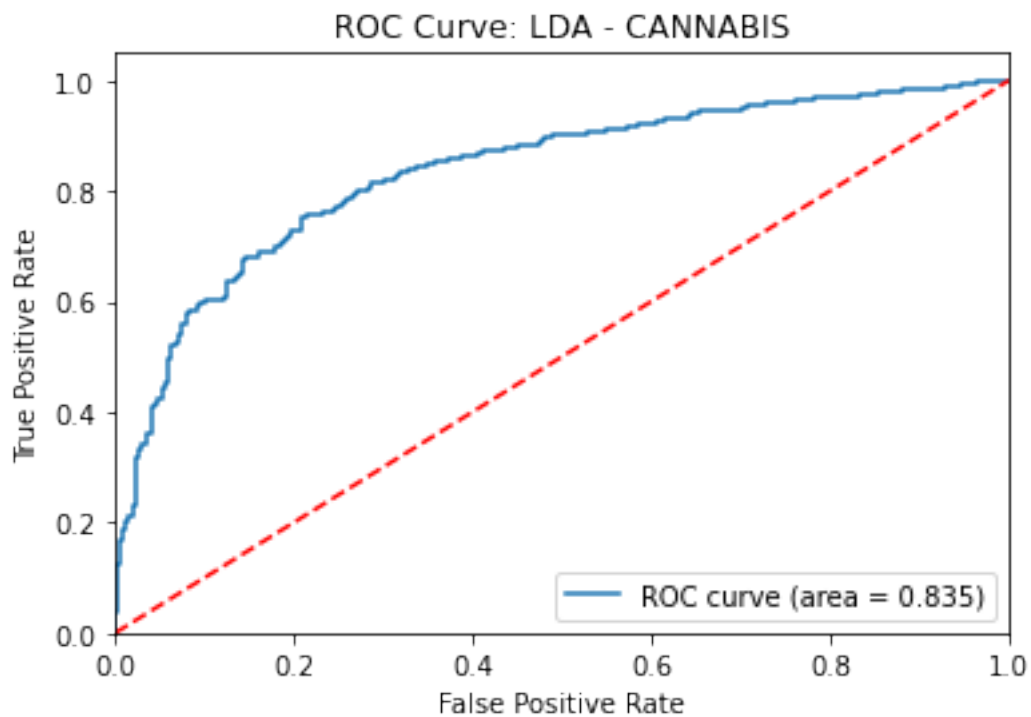


--CANNABIS

Train-Test Split:  
Accuracy: 0.801  
Precision: 0.835  
Recall: 0.924  
F1 score: 0.878  
Confusion matrix:  
[[ 33 53]  
 [ 22 269]]

Cross Validation:  
Mean Accuracy: 0.79841 +/- 0.03645  
Mean Precision: 0.84008 +/- 0.00908  
Mean Recall: 0.93308 +/- 0.02104  
Mean F1 score: 0.87 +/- 0.03

Mean AUC Score: 0.83598 +/- 0.02680



--COKE

Train-Test Split:

Accuracy: 0.653

Precision: 0.573

Recall: 0.562

F1 score: 0.568

Confusion matrix:

```
[[160  64]
```

```
 [ 67  86]]
```

Cross Validation:

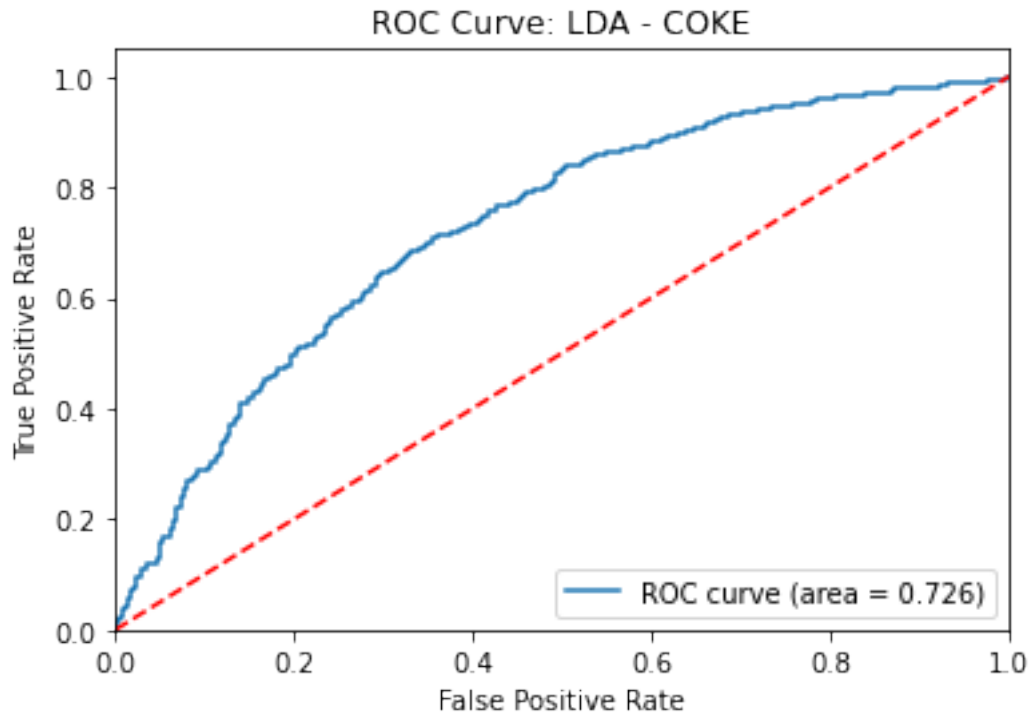
Mean Accuracy: 0.65623 +/- 0.03983

Mean Precision: 0.65020 +/- 0.00955

Mean Recall: 0.60517 +/- 0.04017

Mean F1 score: 0.58 +/- 0.10

Mean AUC Score: 0.72776 +/- 0.01931



--CRACK

Train-Test Split:

Accuracy: 0.886

Precision: 0.500

Recall: 0.047

F1 score: 0.085

Confusion matrix:

```
[[332  2]
```

```
 [ 41  2]]
```

Cross Validation:

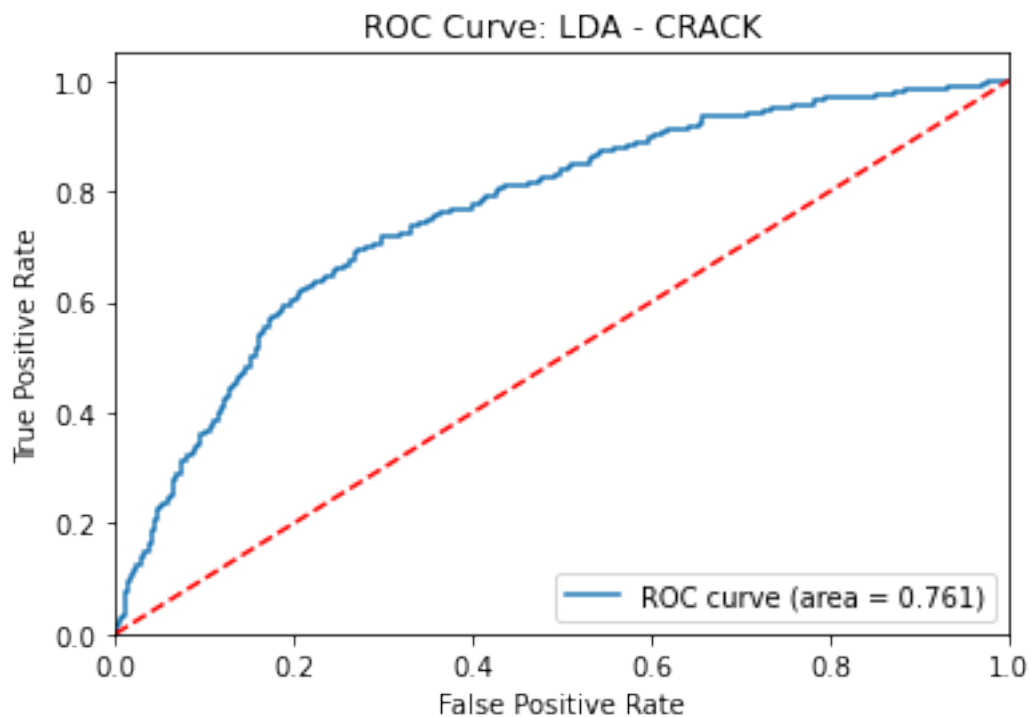
Mean Accuracy: 0.86101 +/- 0.00897

Mean Precision: 0.51278 +/- 0.16473

Mean Recall: 0.07442 +/- 0.02712

Mean F1 score: 0.09 +/- 0.05

Mean AUC Score: 0.76403 +/- 0.04544



--ECSTASY

Train-Test Split:

Accuracy: 0.687

Precision: 0.665

Recall: 0.609

F1 score: 0.636

Confusion matrix:

```
[[156  52]
```

```
 [ 66 103]]
```

Cross Validation:

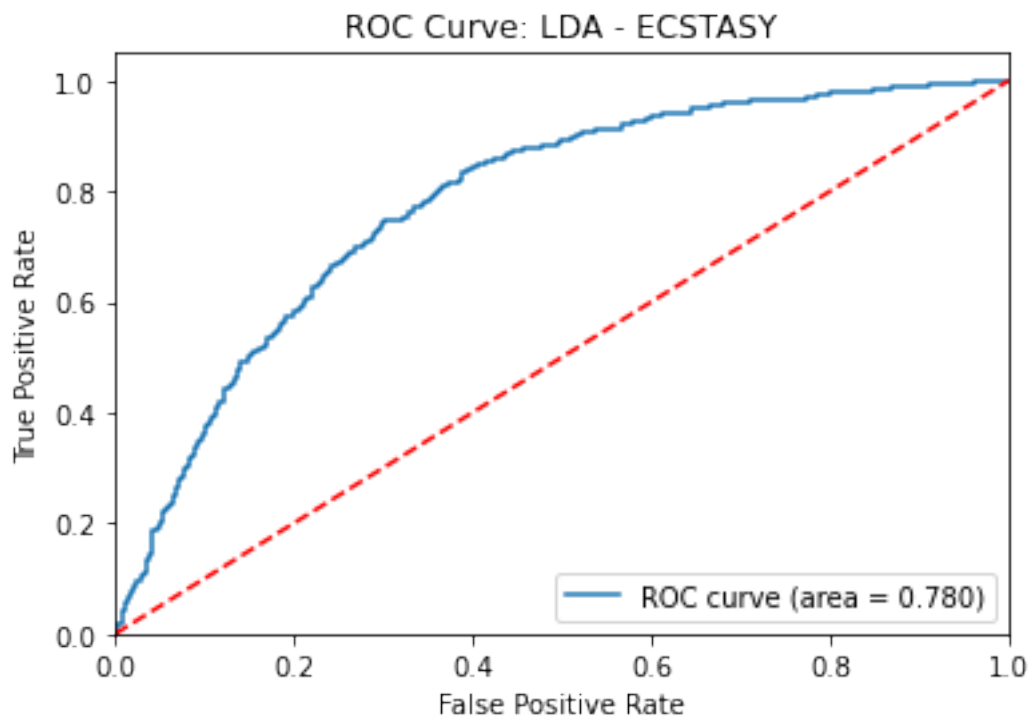
Mean Accuracy: 0.69231 +/- 0.05662

Mean Precision: 0.70329 +/- 0.02946

Mean Recall: 0.65899 +/- 0.04785

Mean F1 score: 0.62 +/- 0.15

Mean AUC Score: 0.78189 +/- 0.01871



--HEROIN

Train-Test Split:

Accuracy: 0.857

Precision: 0.444

Recall: 0.075

F1 score: 0.129

Confusion matrix:

```
[[319  5]
```

```
 [ 49  4]]
```

Cross Validation:

Mean Accuracy: 0.84668 +/- 0.01130

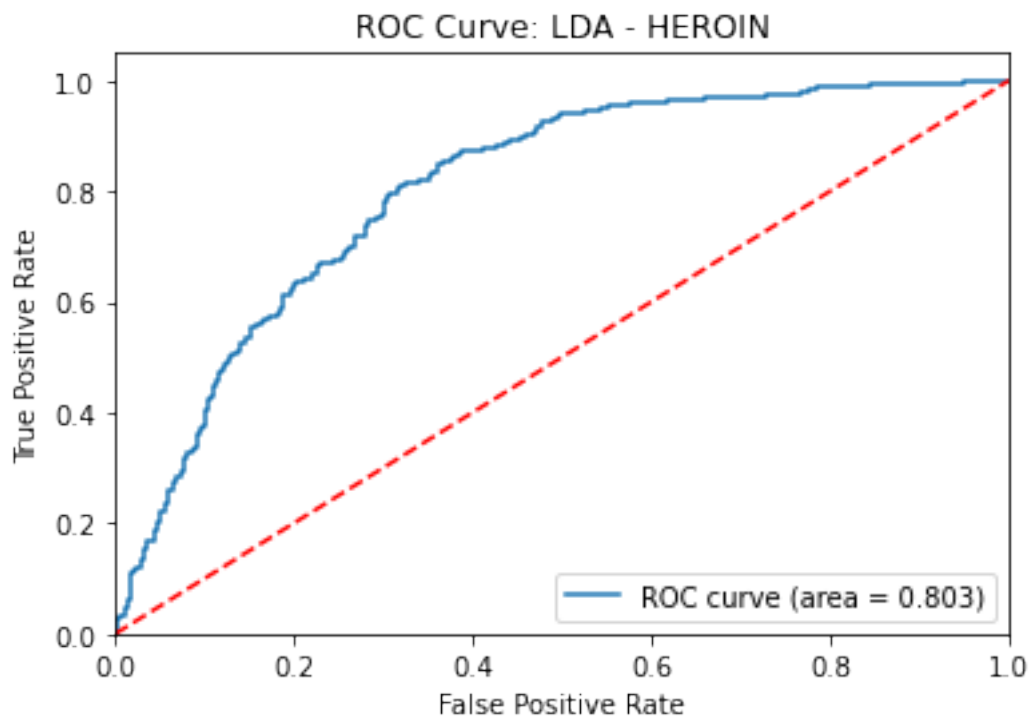
Mean Precision: 0.44096 +/- 0.12979

Mean Recall: 0.13237 +/- 0.05301

Mean F1 score: 0.20 +/- 0.06

Mean AUC Score: 0.80473 +/- 0.03388





--KETAMINE

Train-Test Split:

Accuracy: 0.828

Precision: 0.700

Recall: 0.101

F1 score: 0.177

Confusion matrix:

```
[[305  3]
```

```
 [ 62  7]]
```

Cross Validation:

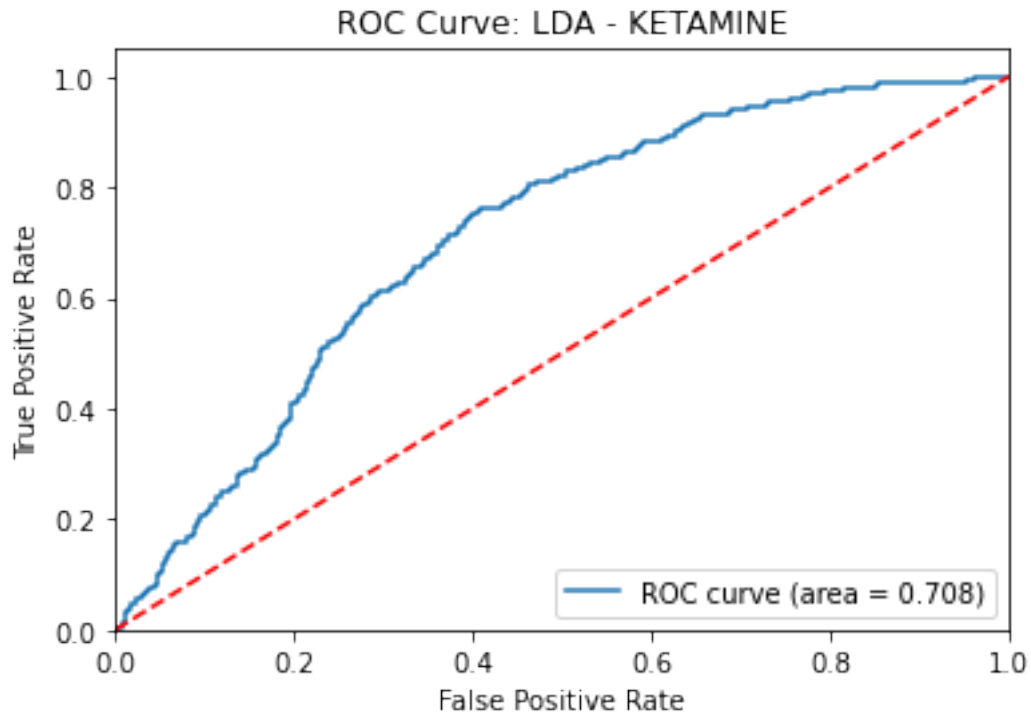
Mean Accuracy: 0.77931 +/- 0.02303

Mean Precision: 0.40857 +/- 0.08310

Mean Recall: 0.05221 +/- 0.01857

Mean F1 score: 0.12 +/- 0.06

Mean AUC Score: 0.70899 +/- 0.02288



--LEGALH

Train-Test Split:

Accuracy: 0.788

Precision: 0.761

Recall: 0.733

F1 score: 0.747

Confusion matrix:

```
[[179 37]
```

```
 [ 43 118]]
```

Cross Validation:

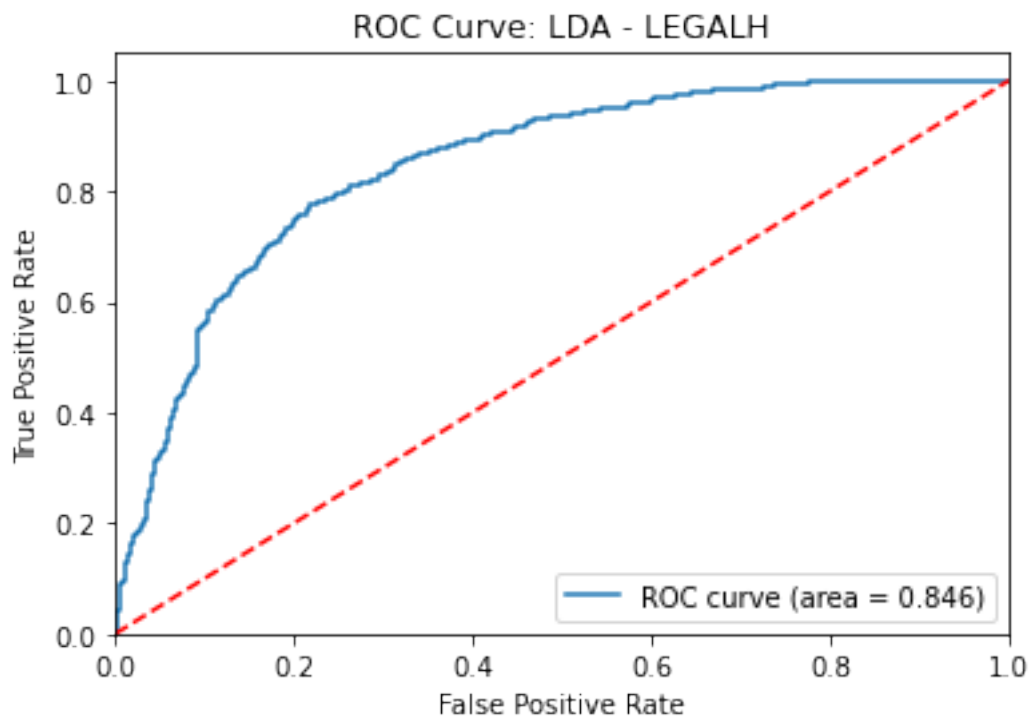
Mean Accuracy: 0.76658 +/- 0.07312

Mean Precision: 0.73962 +/- 0.03643

Mean Recall: 0.71429 +/- 0.04461

Mean F1 score: 0.70 +/- 0.15

Mean AUC Score: 0.84660 +/- 0.01834



--LSD

Train-Test Split:

Accuracy: 0.698

Precision: 0.647

Recall: 0.623

F1 score: 0.635

Confusion matrix:

[[164 54]

[ 60 99]]

Cross Validation:

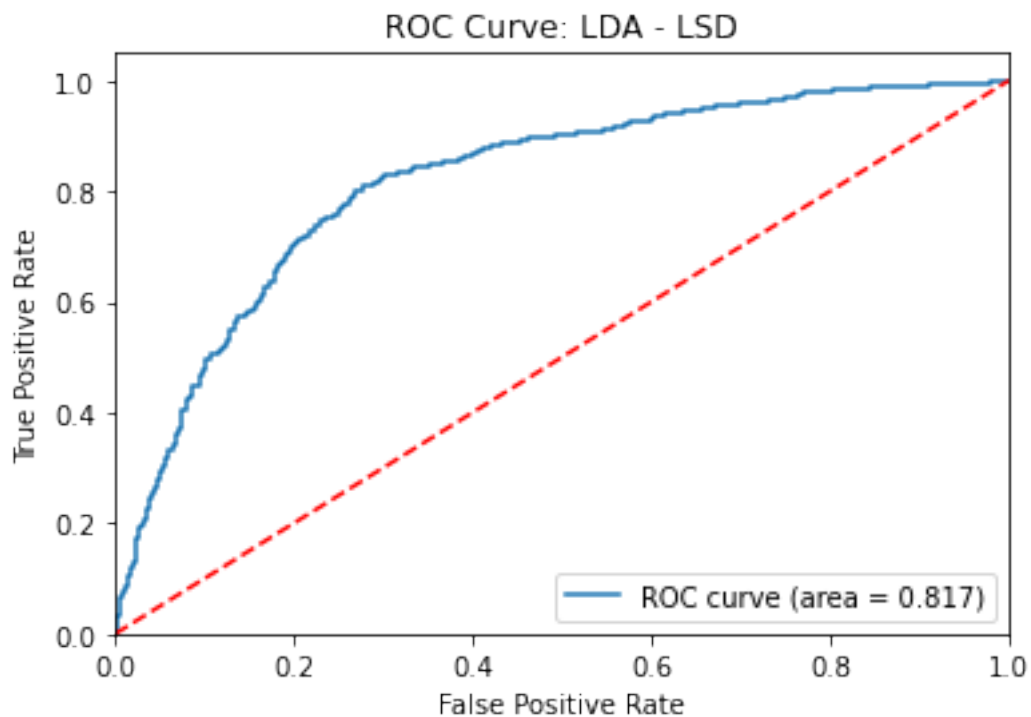
Mean Accuracy: 0.73156 +/- 0.06015

Mean Precision: 0.73006 +/- 0.01995

Mean Recall: 0.69718 +/- 0.02479

Mean F1 score: 0.65 +/- 0.16

Mean AUC Score: 0.81695 +/- 0.01442



--METH

Train-Test Split:

Accuracy: 0.764

Precision: 0.531

Recall: 0.366

F1 score: 0.433

Confusion matrix:

```
[[254 30]
```

```
 [ 59 34]]
```

Cross Validation:

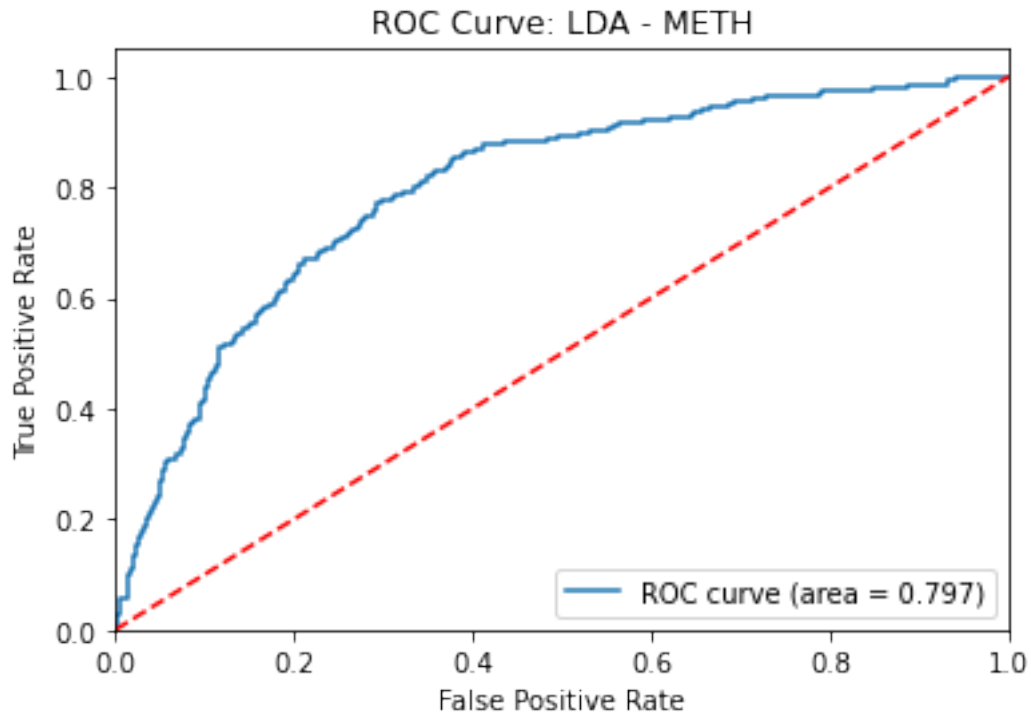
Mean Accuracy: 0.76340 +/- 0.03336

Mean Precision: 0.58012 +/- 0.04820

Mean Recall: 0.46016 +/- 0.02419

Mean F1 score: 0.45 +/- 0.13

Mean AUC Score: 0.79784 +/- 0.01248



--MUSHROOMS

Train-Test Split:

Accuracy: 0.703

Precision: 0.718

Recall: 0.656

F1 score: 0.685

Confusion matrix:

[[143 48]

[ 64 122]]

Cross Validation:

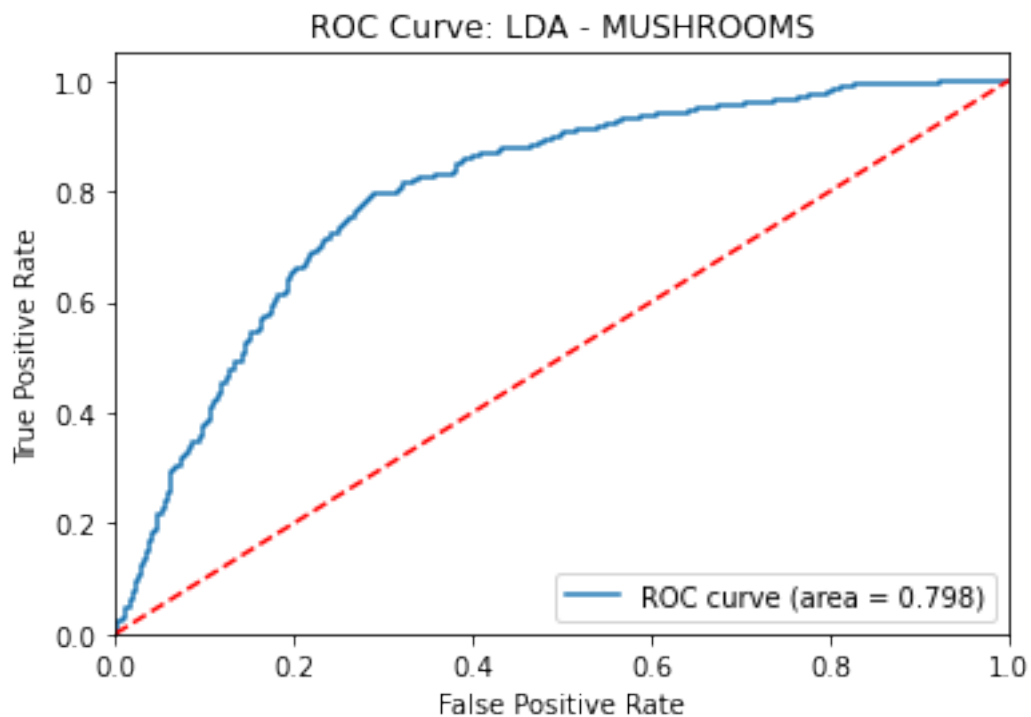
Mean Accuracy: 0.71300 +/- 0.06885

Mean Precision: 0.74014 +/- 0.03287

Mean Recall: 0.68629 +/- 0.05061

Mean F1 score: 0.66 +/- 0.16

Mean AUC Score: 0.80025 +/- 0.02469



--NICOTINE

Train-Test Split:

Accuracy: 0.756

Precision: 0.769

Recall: 0.969

F1 score: 0.858

Confusion matrix:

```
[[ 8 83]
```

```
 [ 9 277]]
```

Cross Validation:

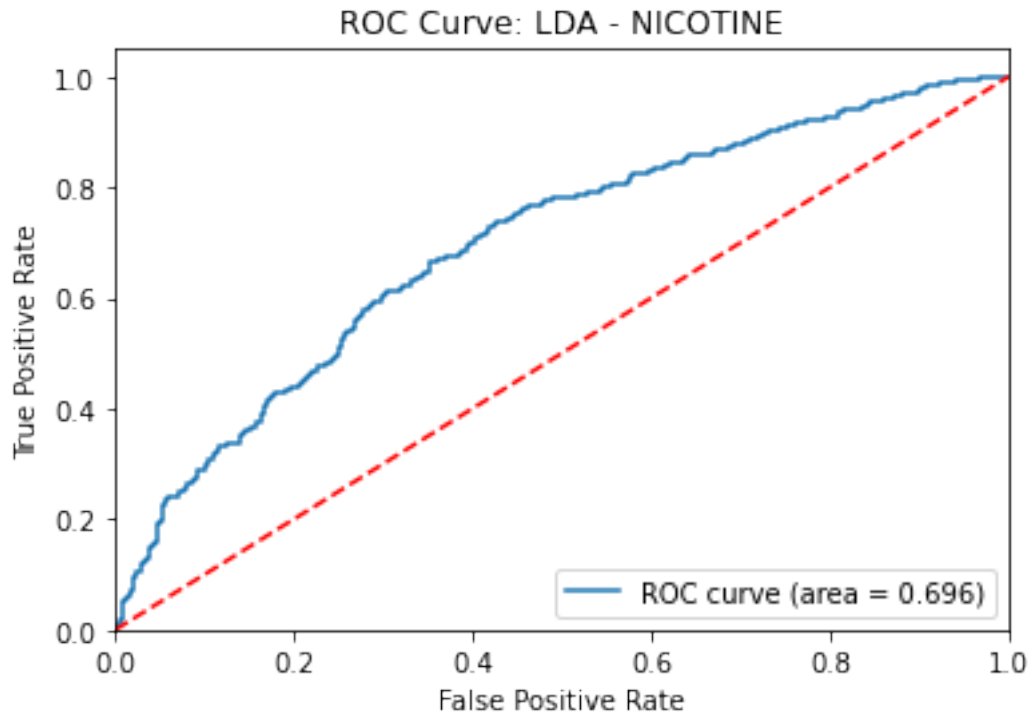
Mean Accuracy: 0.76711 +/- 0.02485

Mean Precision: 0.78983 +/- 0.00398

Mean Recall: 0.97864 +/- 0.00716

Mean F1 score: 0.86 +/- 0.02

Mean AUC Score: 0.69720 +/- 0.02135



## 12.2 K-Nearest Neighbors (KNN)

[7]:

```
df = pd.read_csv('data6.csv')
print("")

for col_index in range(9, len(df.columns)):

    columnName = df.columns[col_index]
    print()
    print("--" + (columnName))
    print()

    print('Train-Test Split:')

    X = df.iloc[:, 1:9]
    y = df[columnName]

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
↳ random_state=42)
```

```

# Create an instance of the model
model_1 = KNeighborsClassifier()

# Train the model on the training data
model_1.fit(X_train, y_train)

# Make predictions on the test data
y_pred1 = model_1.predict(X_test)

# Calculate evaluation metrics
accuracy = accuracy_score(y_test, y_pred1)
precision = precision_score(y_test, y_pred1)
recall = recall_score(y_test, y_pred1)
f1 = f1_score(y_test, y_pred1)
conf_matrix = confusion_matrix(y_test, y_pred1)

# Print the evaluation metrics
print(f"Accuracy: {accuracy:.3f}")
print(f"Precision: {precision:.3f}")
print(f"Recall: {recall:.3f}")
print(f"F1 score: {f1:.3f}")
print("Confusion matrix:")
print(conf_matrix)
print()

N_FOLDS = 5
# Perform cross-validation predictions
y_scores = cross_val_predict(model_1, X_train, y_train, cv=N_FOLDS,
↪method="predict_proba")

fpr, tpr, thresholds = roc_curve(y_train, y_scores[:, 1])
roc_auc = roc_auc_score(y_train, y_scores[:, 1])

print('Cross Validation:')
cv_accuracy = cross_val_score(model_1, X, y, cv=5, scoring='accuracy')
print('Mean Accuracy: %.5f +/- %.5f' % (np.mean(cv_accuracy), np.
↪std(cv_accuracy)))
precision_scores = cross_val_score(model_1, X_train, y_train, cv=N_FOLDS,
↪scoring="precision")
print('Mean Precision: %.5f +/- %.5f' % (np.mean(precision_scores), np.
↪std(precision_scores)))

```



```

    recall_scores = cross_val_score(model_1, X_train, y_train, cv=N_FOLDS,
    ↪scoring="recall")
    cv_f1 = cross_val_score(model_1, X, y, cv=5, scoring='f1')
    print('Mean Recall: %.5f +/- %.5f' % (np.mean(recall_scores), np.
    ↪std(recall_scores)))
    auc_scores = cross_val_score(model_1, X_train, y_train, cv=N_FOLDS,
    ↪scoring="roc_auc")
    print(f"Mean F1 score: {np.mean(cv_f1):.2f} +/- {np.std(cv_f1):.2f}")
    print('Mean AUC Score: %.5f +/- %.5f' % (np.mean(auc_scores), np.
    ↪std(auc_scores)))
    print()
    print()

    # Plot ROC curve
    plt.figure()
    plt.plot(fpr, tpr, label='ROC curve (area = %0.3f)' % roc_auc)
    plt.plot([0, 1], [0, 1], 'r--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('ROC Curve: KNN - ' + (columnName))
    plt.legend(loc="lower right")
    plt.show()

```

--AMPHET

Train-Test Split:

Accuracy: 0.613

Precision: 0.598

Recall: 0.591

F1 score: 0.594

Confusion matrix:

```
[[124  72]
```

```
 [ 74 107]]
```

Cross Validation:

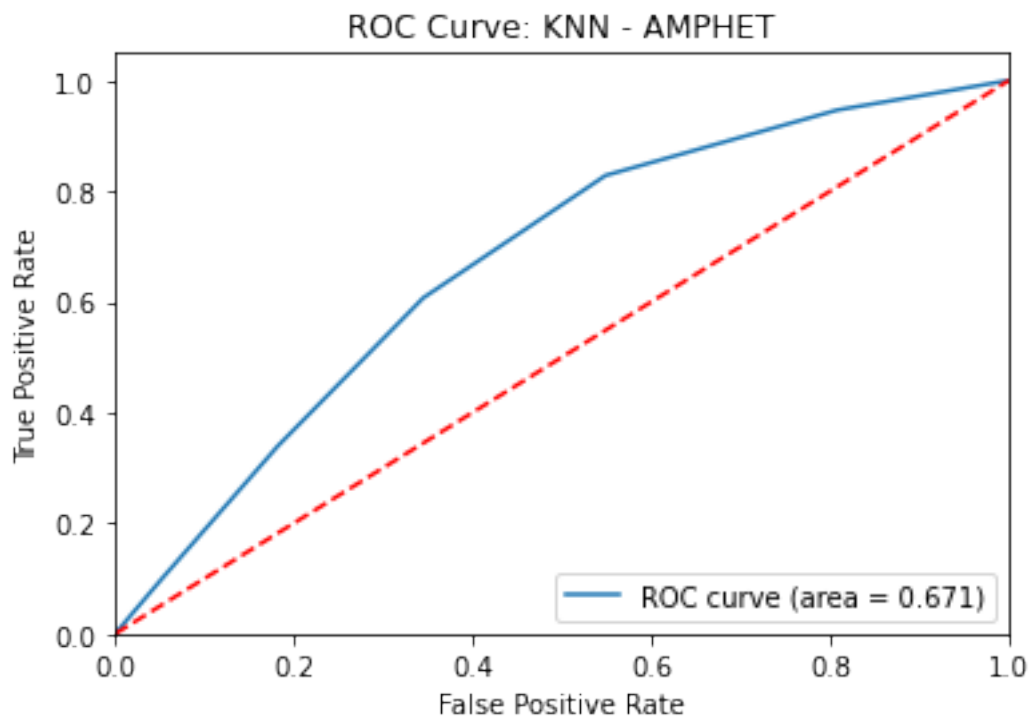
Mean Accuracy: 0.61220 +/- 0.02829

Mean Precision: 0.62209 +/- 0.01341

Mean Recall: 0.60712 +/- 0.02606

Mean F1 score: 0.59 +/- 0.06

Mean AUC Score: 0.67218 +/- 0.01040



--AMYL

Train-Test Split:

Accuracy: 0.716

Precision: 0.562

Recall: 0.353

F1 score: 0.434

Confusion matrix:

```
[[229 32]
```

```
 [ 75 41]]
```

Cross Validation:

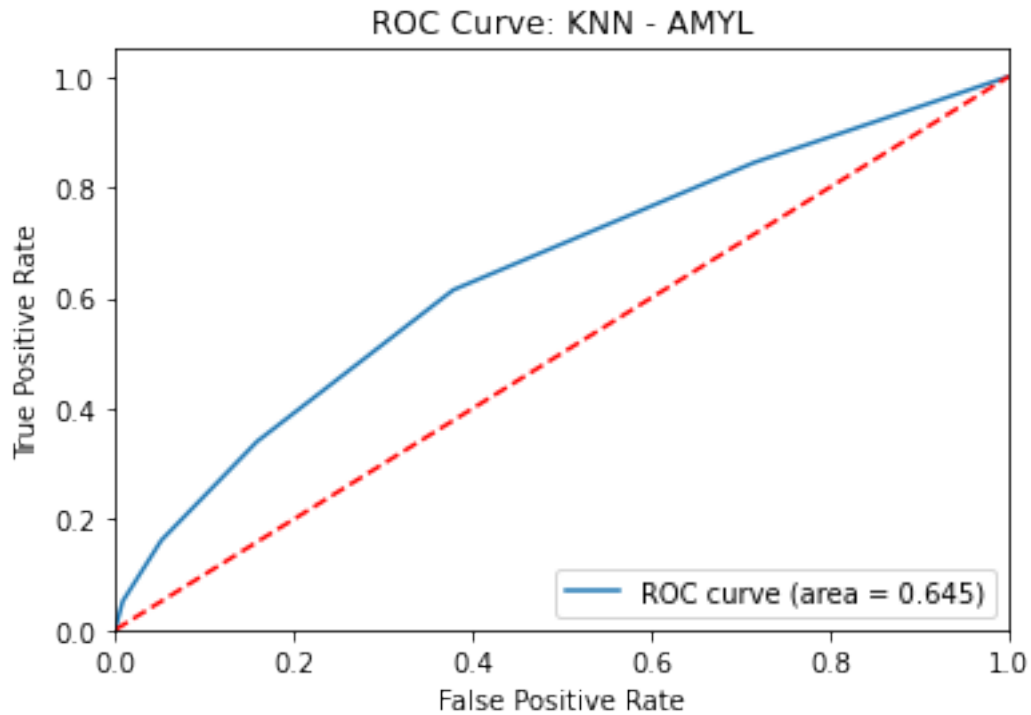
Mean Accuracy: 0.68647 +/- 0.02297

Mean Precision: 0.49085 +/- 0.04132

Mean Recall: 0.34049 +/- 0.02268

Mean F1 score: 0.41 +/- 0.06

Mean AUC Score: 0.64538 +/- 0.02496



--BENZOS

Train-Test Split:

Accuracy: 0.684

Precision: 0.665

Recall: 0.638

F1 score: 0.651

Confusion matrix:

```
[[147  56]
```

```
 [ 63 111]]
```

Cross Validation:

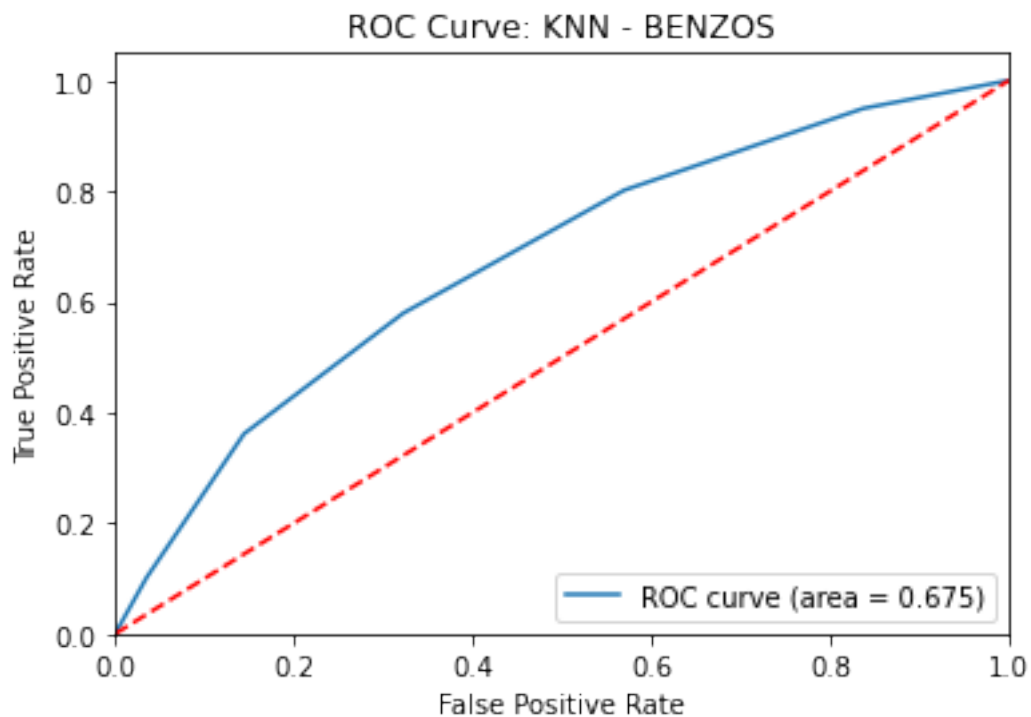
Mean Accuracy: 0.63660 +/- 0.01600

Mean Precision: 0.61578 +/- 0.01844

Mean Recall: 0.57949 +/- 0.02748

Mean F1 score: 0.60 +/- 0.05

Mean AUC Score: 0.67494 +/- 0.02263



--CANNABIS

Train-Test Split:

Accuracy: 0.782

Precision: 0.836

Recall: 0.893

F1 score: 0.864

Confusion matrix:

```
[[ 35  51]
```

```
 [ 31 260]]
```

Cross Validation:

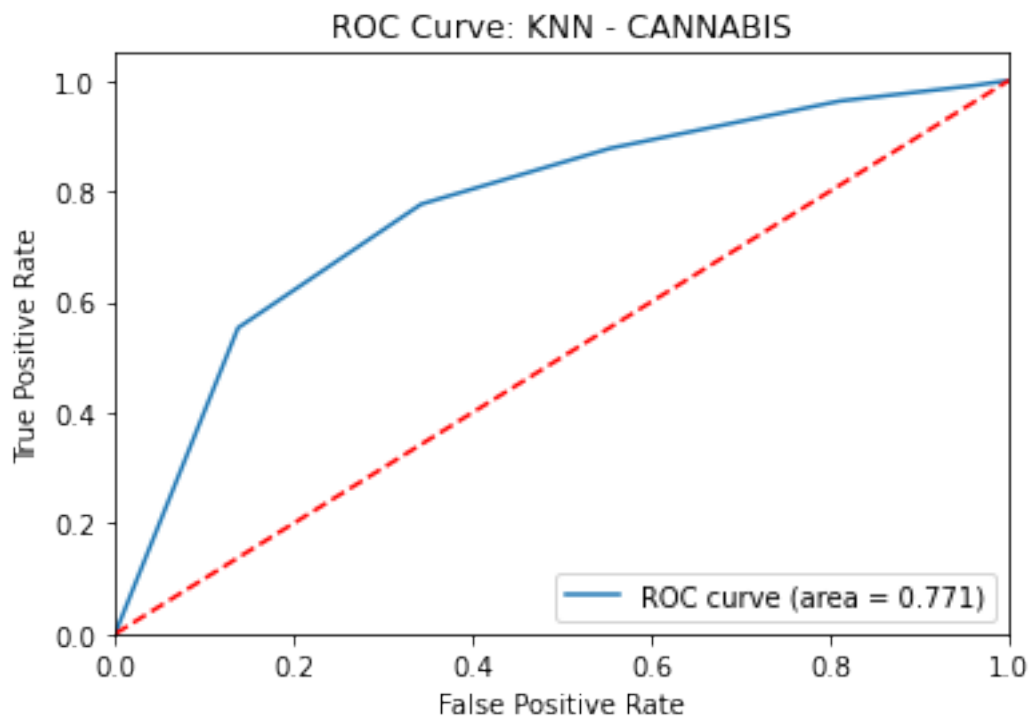
Mean Accuracy: 0.77772 +/- 0.05266

Mean Precision: 0.85137 +/- 0.01012

Mean Recall: 0.87718 +/- 0.03101

Mean F1 score: 0.86 +/- 0.05

Mean AUC Score: 0.77106 +/- 0.02064



--COKE

Train-Test Split:

Accuracy: 0.629

Precision: 0.542

Recall: 0.542

F1 score: 0.542

Confusion matrix:

[[154 70]

[ 70 83]]

Cross Validation:

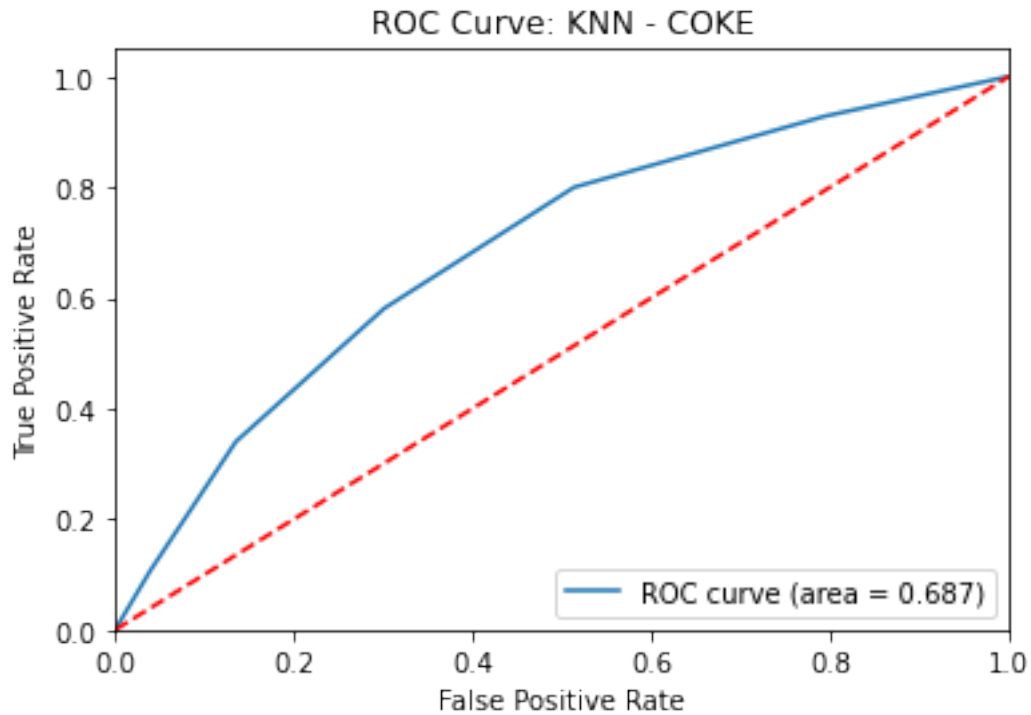
Mean Accuracy: 0.63873 +/- 0.04128

Mean Precision: 0.62166 +/- 0.01978

Mean Recall: 0.58065 +/- 0.03349

Mean F1 score: 0.58 +/- 0.05

Mean AUC Score: 0.68813 +/- 0.01902



--CRACK

Train-Test Split:

Accuracy: 0.867

Precision: 0.316

Recall: 0.140

F1 score: 0.194

Confusion matrix:

```
[[321  13]
```

```
 [ 37   6]]
```

Cross Validation:

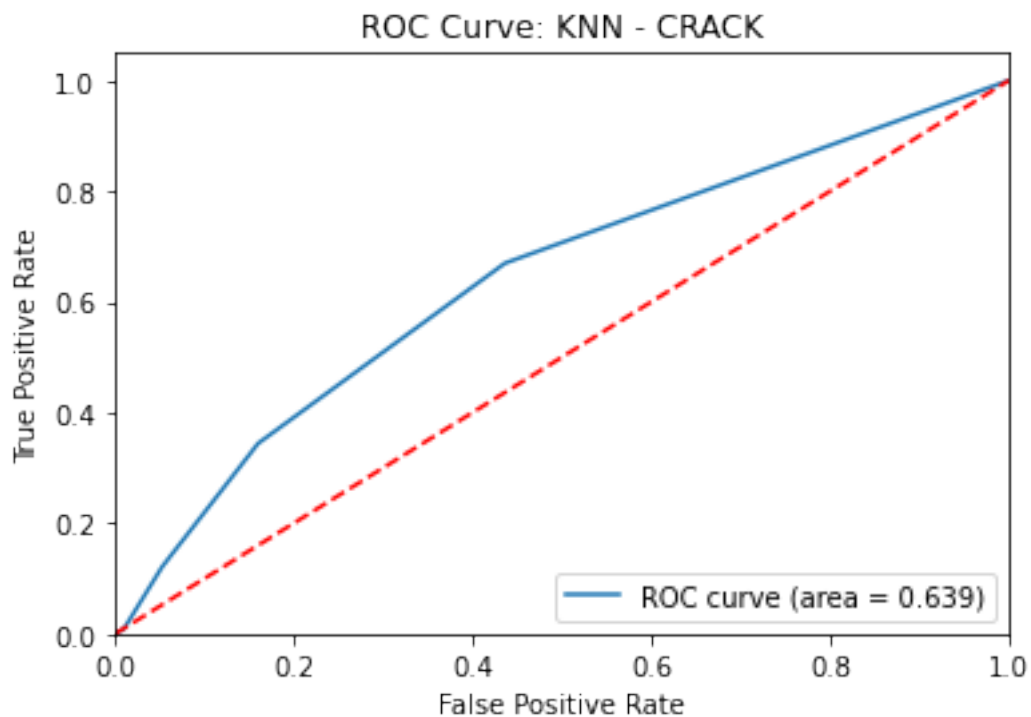
Mean Accuracy: 0.83767 +/- 0.01952

Mean Precision: 0.27942 +/- 0.09333

Mean Recall: 0.12093 +/- 0.04001

Mean F1 score: 0.15 +/- 0.03

Mean AUC Score: 0.63924 +/- 0.01939



--ECSTASY

Train-Test Split:

Accuracy: 0.671

Precision: 0.638

Recall: 0.615

F1 score: 0.627

Confusion matrix:

```
[[149  59]
```

```
 [ 65 104]]
```

Cross Validation:

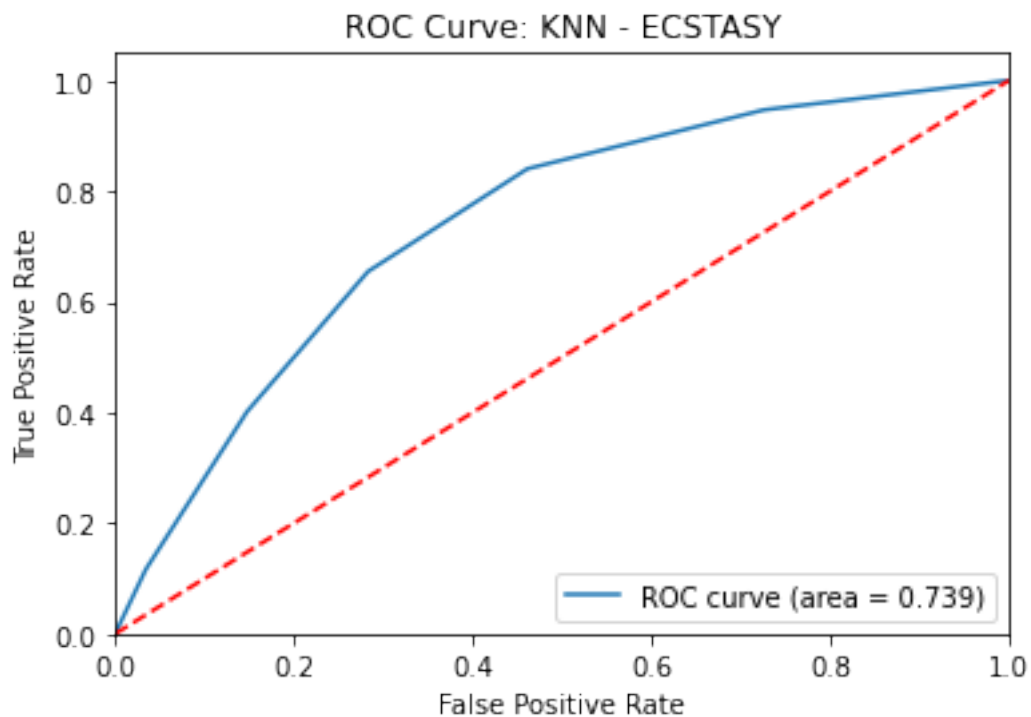
Mean Accuracy: 0.68064 +/- 0.04351

Mean Precision: 0.66690 +/- 0.02847

Mean Recall: 0.65468 +/- 0.04220

Mean F1 score: 0.65 +/- 0.07

Mean AUC Score: 0.74000 +/- 0.02067



--HEROIN

Train-Test Split:

Accuracy: 0.841

Precision: 0.360

Recall: 0.170

F1 score: 0.231

Confusion matrix:

```
[[308 16]
```

```
 [ 44  9]]
```

Cross Validation:

Mean Accuracy: 0.82653 +/- 0.02242

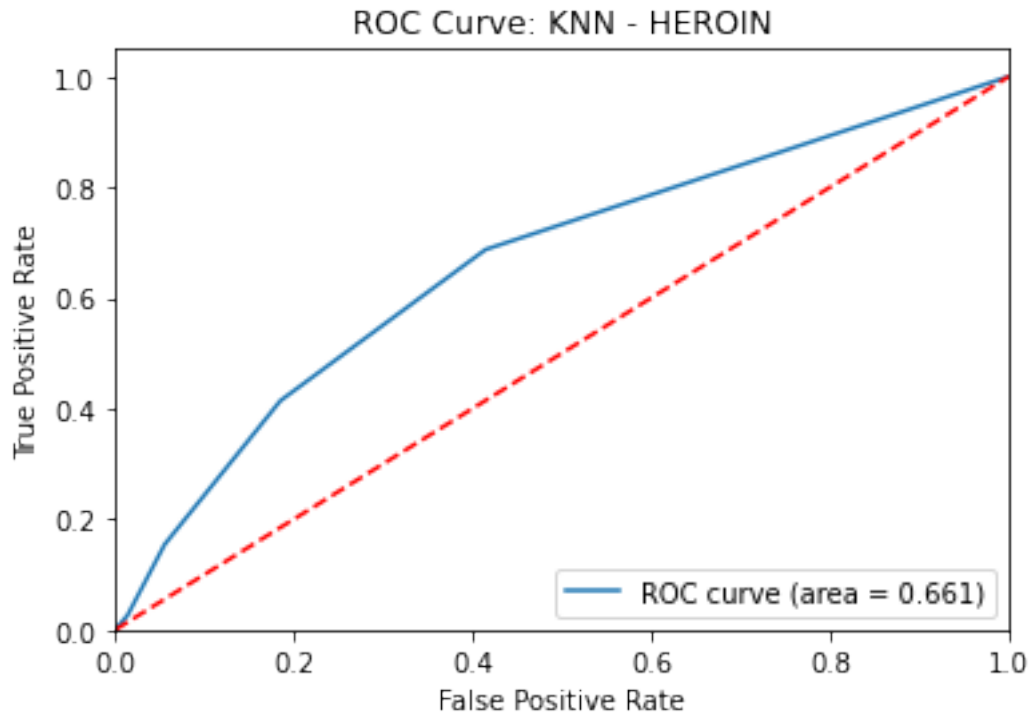
Mean Precision: 0.33955 +/- 0.06631

Mean Recall: 0.15459 +/- 0.04073

Mean F1 score: 0.22 +/- 0.05

Mean AUC Score: 0.66064 +/- 0.04197





--KETAMINE

Train-Test Split:

Accuracy: 0.756

Precision: 0.244

Recall: 0.159

F1 score: 0.193

Confusion matrix:

```
[[274  34]
```

```
 [ 58  11]]
```

Cross Validation:

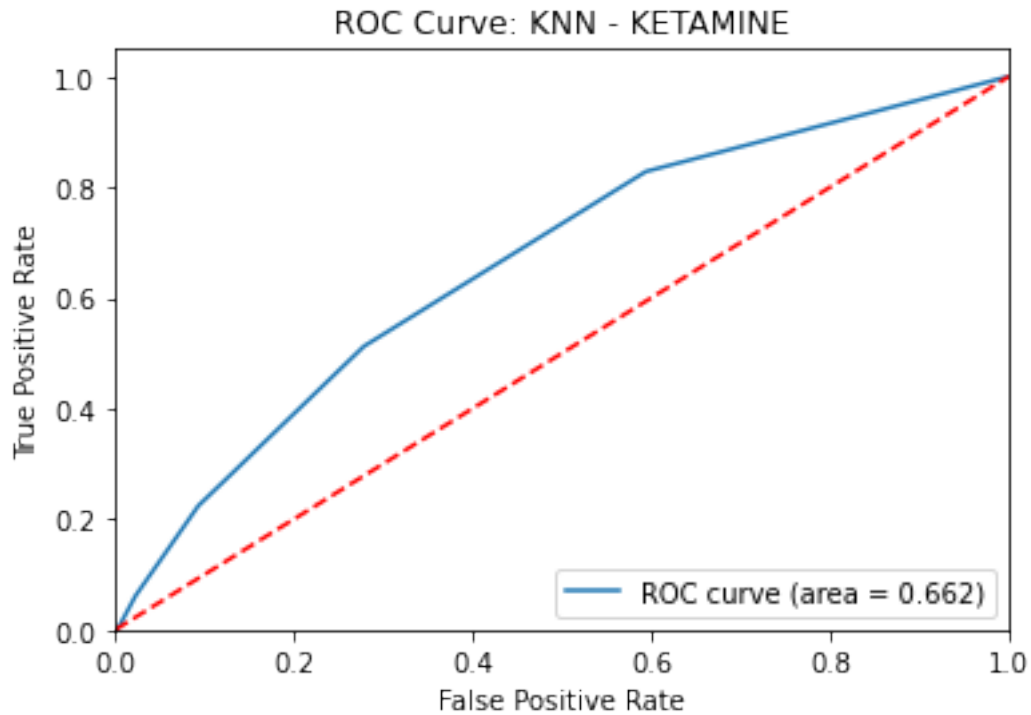
Mean Accuracy: 0.74377 +/- 0.04380

Mean Precision: 0.40260 +/- 0.05585

Mean Recall: 0.22387 +/- 0.04161

Mean F1 score: 0.23 +/- 0.04

Mean AUC Score: 0.66156 +/- 0.02876



--LEGALH

Train-Test Split:

Accuracy: 0.751

Precision: 0.696

Recall: 0.739

F1 score: 0.717

Confusion matrix:

```
[[164  52]
```

```
 [ 42 119]]
```

Cross Validation:

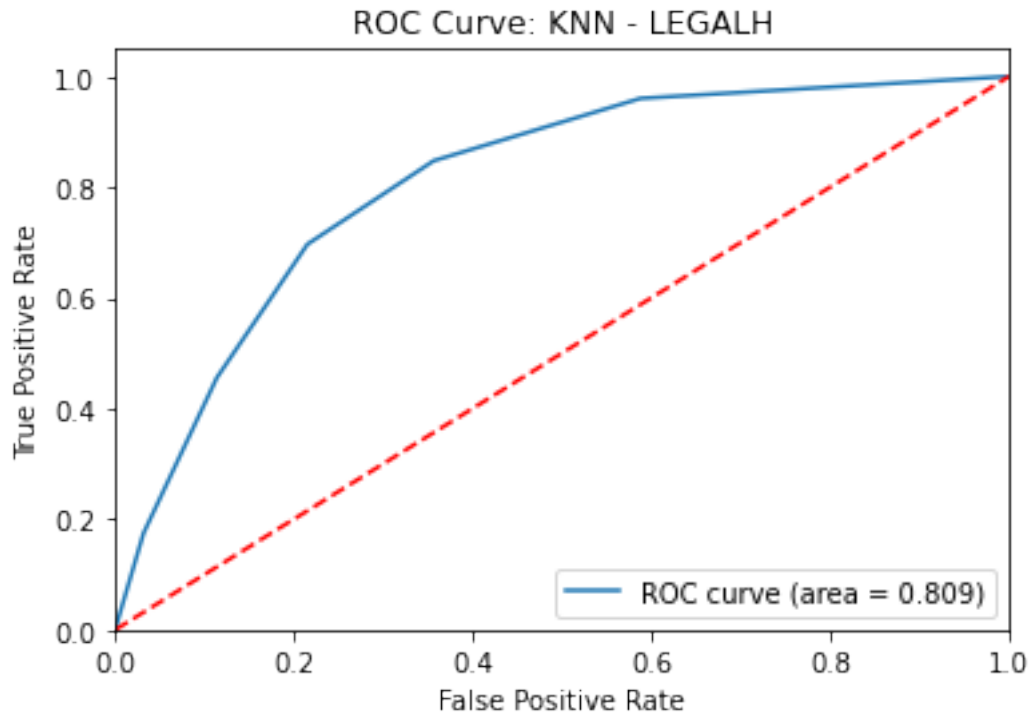
Mean Accuracy: 0.74430 +/- 0.05440

Mean Precision: 0.70145 +/- 0.03962

Mean Recall: 0.69683 +/- 0.02813

Mean F1 score: 0.69 +/- 0.09

Mean AUC Score: 0.80874 +/- 0.02354



--LSD

Train-Test Split:

Accuracy: 0.700

Precision: 0.644

Recall: 0.648

F1 score: 0.646

Confusion matrix:

```
[[161  57]
```

```
 [ 56 103]]
```

Cross Validation:

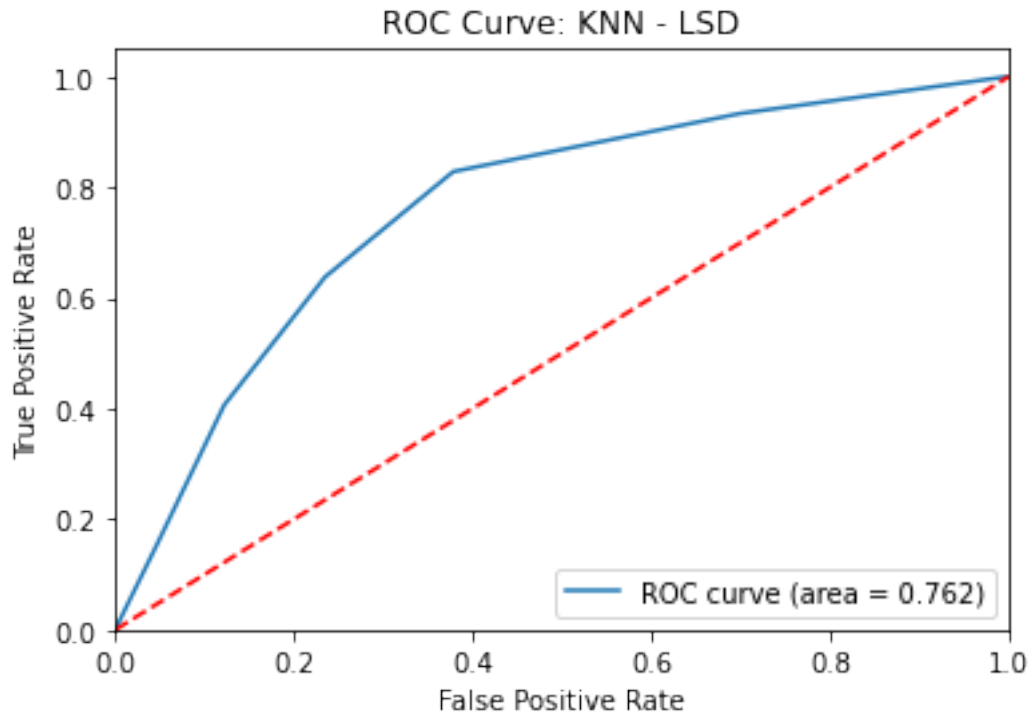
Mean Accuracy: 0.70345 +/- 0.04295

Mean Precision: 0.67739 +/- 0.02114

Mean Recall: 0.63777 +/- 0.02700

Mean F1 score: 0.64 +/- 0.09

Mean AUC Score: 0.76163 +/- 0.00736



--METH

Train-Test Split:

Accuracy: 0.759

Precision: 0.514

Recall: 0.387

F1 score: 0.442

Confusion matrix:

```
[[250  34]
```

```
 [ 57  36]]
```

Cross Validation:

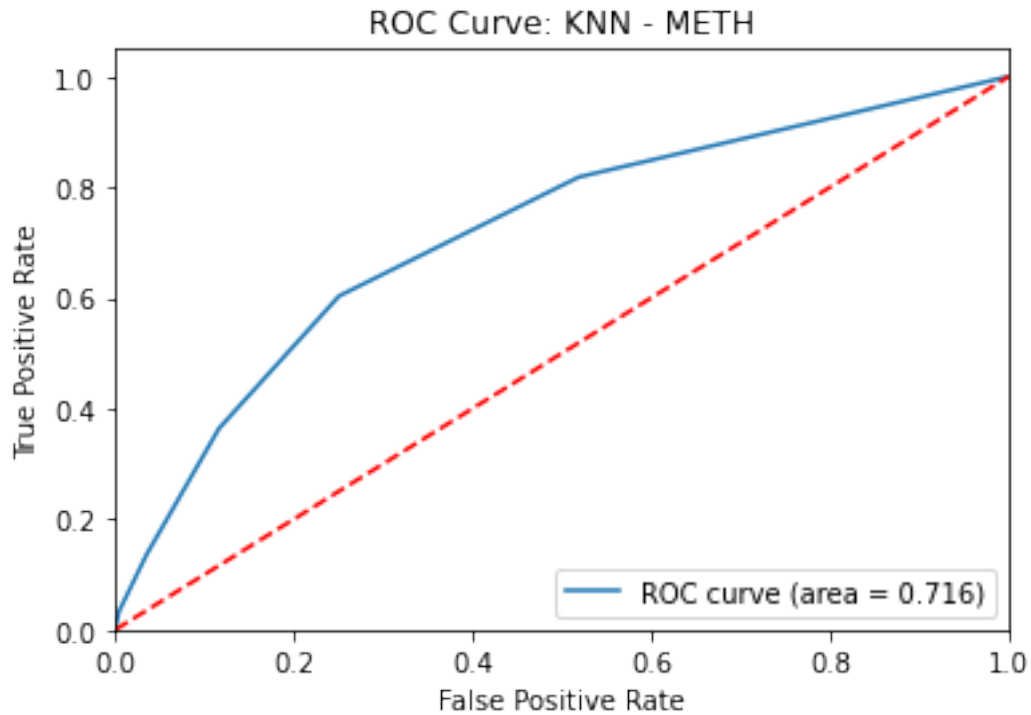
Mean Accuracy: 0.75119 +/- 0.04465

Mean Precision: 0.50050 +/- 0.08258

Mean Recall: 0.36351 +/- 0.05495

Mean F1 score: 0.41 +/- 0.09

Mean AUC Score: 0.71719 +/- 0.01203



--MUSHROOMS

Train-Test Split:

Accuracy: 0.687

Precision: 0.685

Recall: 0.677

F1 score: 0.681

Confusion matrix:

```
[[133  58]
```

```
 [ 60 126]]
```

Cross Validation:

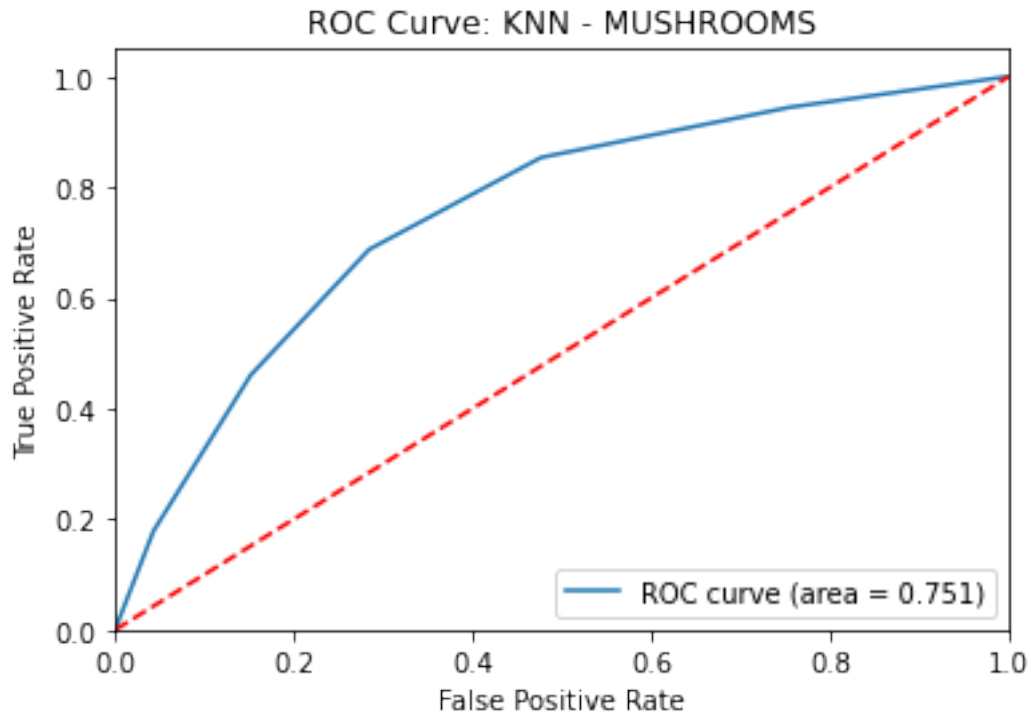
Mean Accuracy: 0.69390 +/- 0.05243

Mean Precision: 0.68686 +/- 0.01542

Mean Recall: 0.68760 +/- 0.01927

Mean F1 score: 0.67 +/- 0.09

Mean AUC Score: 0.75157 +/- 0.01500



--NICOTINE

Train-Test Split:

Accuracy: 0.719

Precision: 0.778

Recall: 0.881

F1 score: 0.826

Confusion matrix:

```
[[ 19  72]
```

```
 [ 34 252]]
```

Cross Validation:

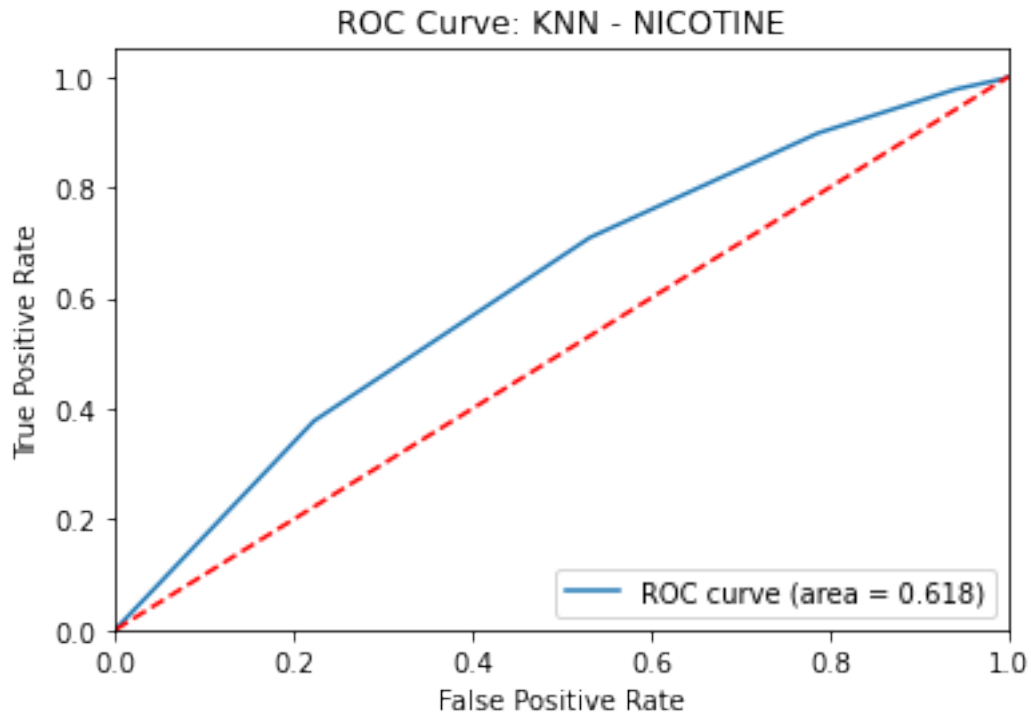
Mean Accuracy: 0.74111 +/- 0.04189

Mean Precision: 0.79859 +/- 0.00769

Mean Recall: 0.89753 +/- 0.01851

Mean F1 score: 0.84 +/- 0.03

Mean AUC Score: 0.61871 +/- 0.02169



### 12.3 Decision Tree (CART)

[9]:

```
df = pd.read_csv('data6.csv')
print("")

for col_index in range(9, len(df.columns)):

    columnName = df.columns[col_index]
    print()
    print("--" + (columnName))
    print()

    print('Train-Test Split:')

    X = df.iloc[:, 1:9]
    y = df[columnName]
```

```

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
↪random_state=42)

# Create an instance of the model
model_1 = DecisionTreeClassifier()

# Train the model on the training data
model_1.fit(X_train, y_train)

# Make predictions on the test data
y_pred1 = model_1.predict(X_test)

# Calculate evaluation metrics
accuracy = accuracy_score(y_test, y_pred1)
precision = precision_score(y_test, y_pred1)
recall = recall_score(y_test, y_pred1)
f1 = f1_score(y_test, y_pred1)
conf_matrix = confusion_matrix(y_test, y_pred1)

# Print the evaluation metrics
print(f"Accuracy: {accuracy:.3f}")
print(f"Precision: {precision:.3f}")
print(f"Recall: {recall:.3f}")
print(f"F1 score: {f1:.3f}")
print("Confusion matrix:")
print(conf_matrix)
print()

N_FOLDS = 5
# Perform cross-validation predictions
y_scores = cross_val_predict(model_1, X_train, y_train, cv=N_FOLDS,
↪method="predict_proba")

fpr, tpr, thresholds = roc_curve(y_train, y_scores[:, 1])
roc_auc = roc_auc_score(y_train, y_scores[:, 1])

print('Cross Validation:')
cv_accuracy = cross_val_score(model_1, X, y, cv=5, scoring='accuracy')
print('Mean Accuracy: %.5f +/- %.5f' % (np.mean(cv_accuracy), np.
↪std(cv_accuracy)))
precision_scores = cross_val_score(model_1, X_train, y_train, cv=N_FOLDS,
↪scoring="precision")

```



```

    print('Mean Precision: %.5f +/- %.5f' % (np.mean(precision_scores), np.
↪std(precision_scores)))
    recall_scores = cross_val_score(model_1, X_train, y_train, cv=N_FOLDS,
↪scoring="recall")
    cv_f1 = cross_val_score(model_1, X, y, cv=5, scoring='f1')
    print(f"Mean F1 score: {np.mean(cv_f1):.2f} +/- {np.std(cv_f1):.2f}")
    print('Mean Recall: %.5f +/- %.5f' % (np.mean(recall_scores), np.
↪std(recall_scores)))
    auc_scores = cross_val_score(model_1, X_train, y_train, cv=N_FOLDS,
↪scoring="roc_auc")
    print('Mean AUC Score: %.5f +/- %.5f' % (np.mean(auc_scores), np.
↪std(auc_scores)))
    print()
    print()

    # Plot ROC curve
    plt.figure()
    plt.plot(fpr, tpr, label='ROC curve (area = %0.3f)' % roc_auc)
    plt.plot([0, 1], [0, 1], 'r--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    # Model
    plt.title('ROC Curve: CART - ' + (columnName))
    plt.legend(loc="lower right")
    plt.show()

```

--AMPHET

Train-Test Split:

Accuracy: 0.578

Precision: 0.561

Recall: 0.558

F1 score: 0.560

Confusion matrix:

```
[[117  79]
```

```
 [ 80 101]]
```

Cross Validation:

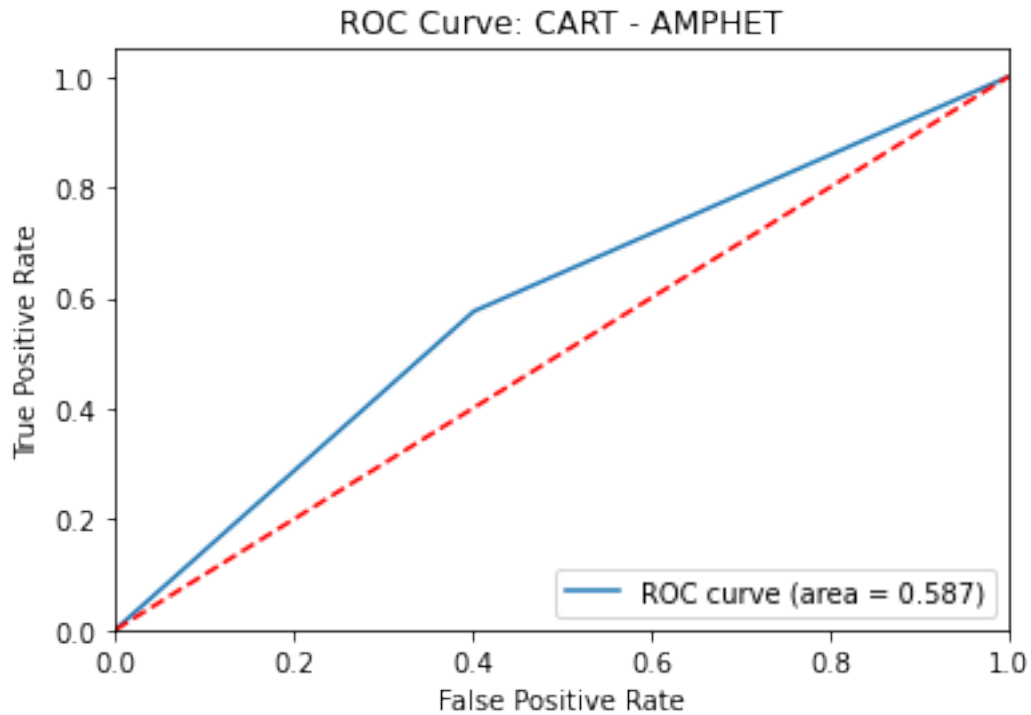
Mean Accuracy: 0.58090 +/- 0.03343

Mean Precision: 0.57685 +/- 0.03739

Mean F1 score: 0.57 +/- 0.04

Mean Recall: 0.57974 +/- 0.06353

Mean AUC Score: 0.58367 +/- 0.03649



--AMYL

Train-Test Split:

Accuracy: 0.615

Precision: 0.374

Recall: 0.371

F1 score: 0.372

Confusion matrix:

```
[[189  72]
```

```
 [ 73  43]]
```

Cross Validation:

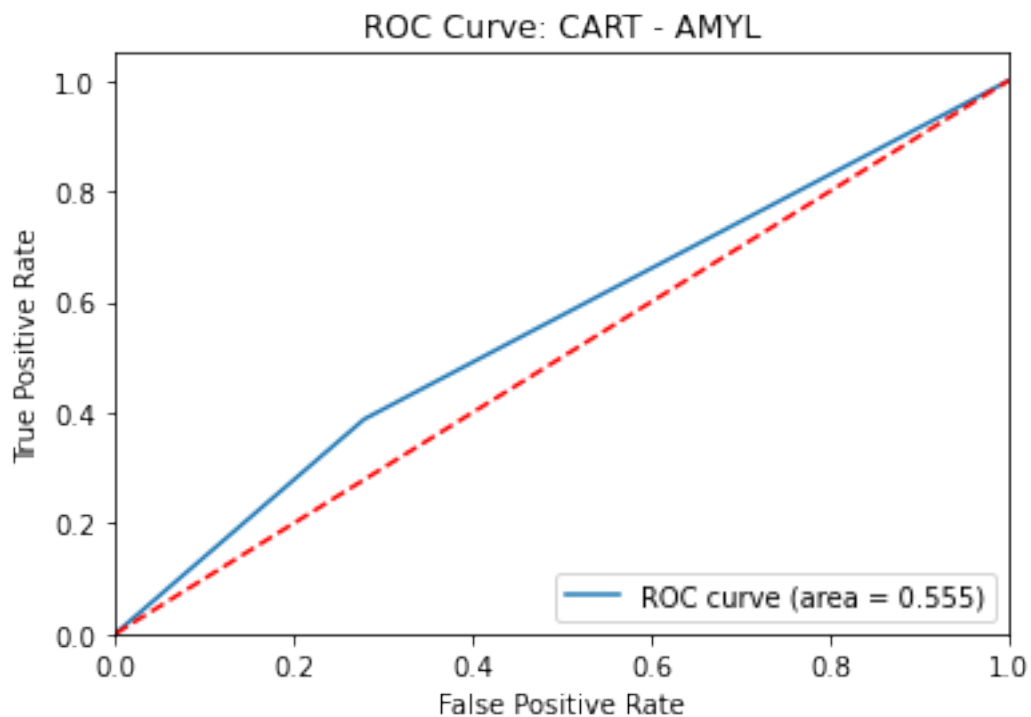
Mean Accuracy: 0.62865 +/- 0.01299

Mean Precision: 0.39405 +/- 0.03813

Mean F1 score: 0.43 +/- 0.04

Mean Recall: 0.40947 +/- 0.03641

Mean AUC Score: 0.56476 +/- 0.02291



--BENZOS

Train-Test Split:

Accuracy: 0.634

Precision: 0.607

Recall: 0.586

F1 score: 0.596

Confusion matrix:

```
[[137  66]
```

```
 [ 72 102]]
```

Cross Validation:

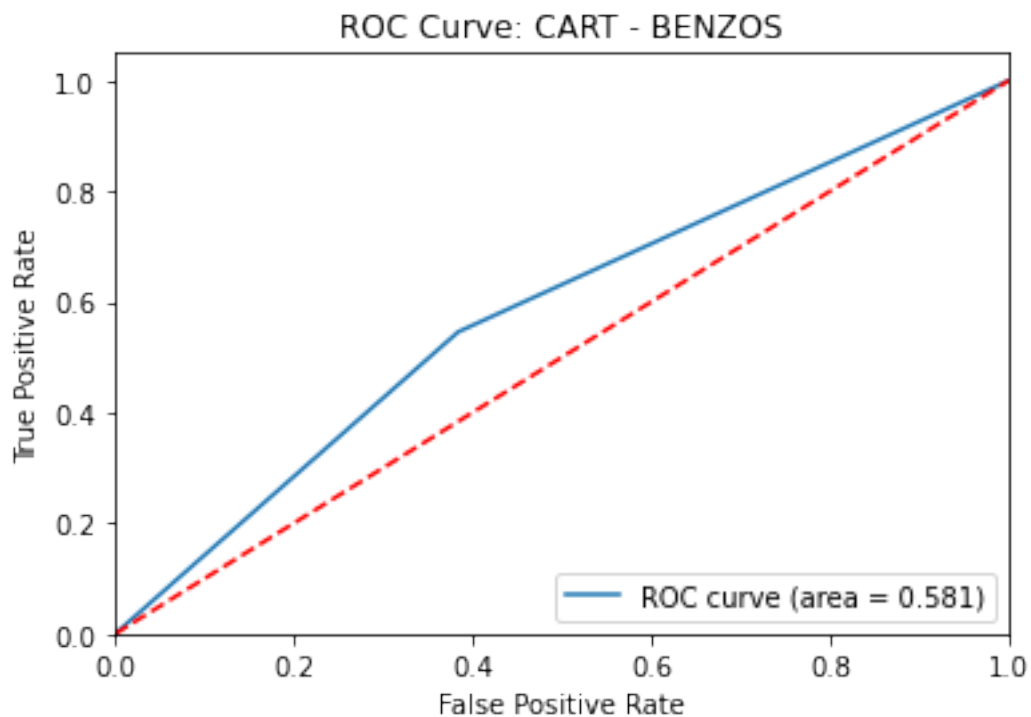
Mean Accuracy: 0.58515 +/- 0.02388

Mean Precision: 0.55350 +/- 0.00808

Mean F1 score: 0.56 +/- 0.03

Mean Recall: 0.54575 +/- 0.03167

Mean AUC Score: 0.59092 +/- 0.01315



--CANNABIS

Train-Test Split:

Accuracy: 0.735

Precision: 0.822

Recall: 0.838

F1 score: 0.830

Confusion matrix:

```
[[ 33  53]
```

```
 [ 47 244]]
```

Cross Validation:

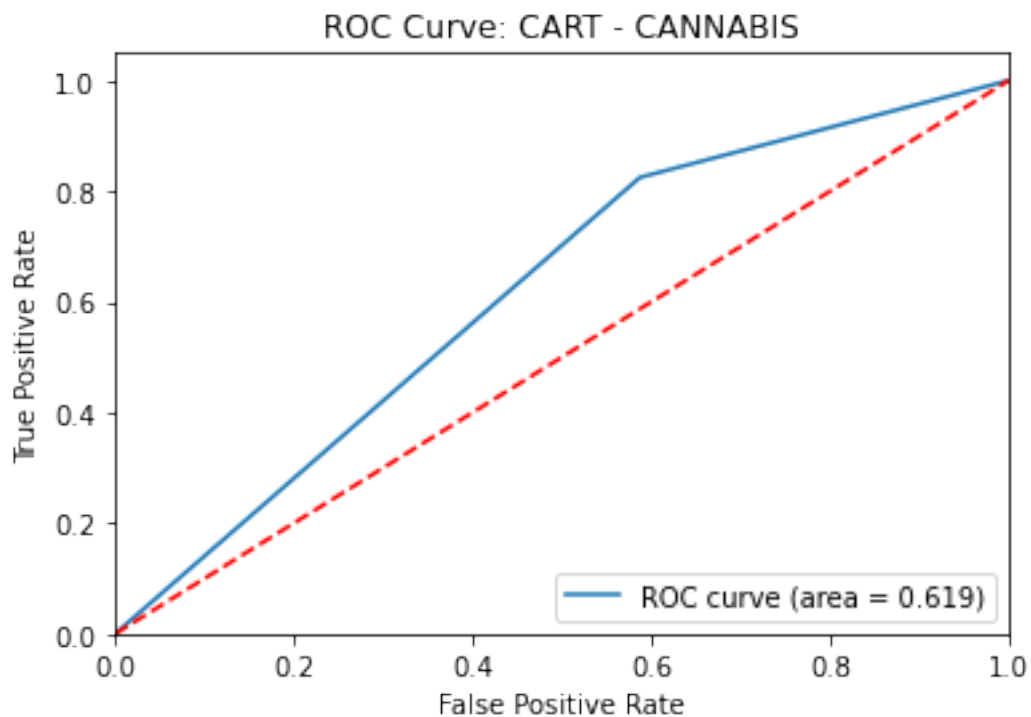
Mean Accuracy: 0.72308 +/- 0.05383

Mean Precision: 0.83527 +/- 0.00786

Mean F1 score: 0.81 +/- 0.05

Mean Recall: 0.82048 +/- 0.01761

Mean AUC Score: 0.62450 +/- 0.03100



--COKE

Train-Test Split:

Accuracy: 0.621

Precision: 0.531

Recall: 0.562

F1 score: 0.546

Confusion matrix:

```
[[148  76]
```

```
 [ 67  86]]
```

Cross Validation:

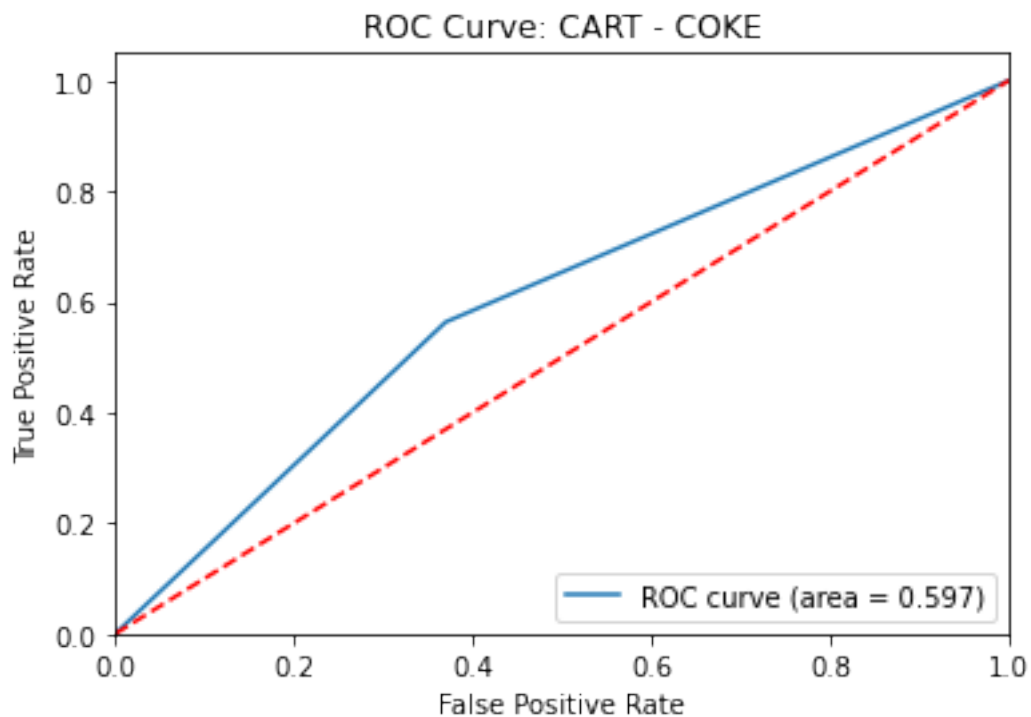
Mean Accuracy: 0.58674 +/- 0.04135

Mean Precision: 0.55134 +/- 0.01981

Mean F1 score: 0.55 +/- 0.06

Mean Recall: 0.55332 +/- 0.02165

Mean AUC Score: 0.58019 +/- 0.01565



--CRACK

Train-Test Split:

Accuracy: 0.775

Precision: 0.150

Recall: 0.209

F1 score: 0.175

Confusion matrix:

```
[[283  51]
```

```
 [ 34   9]]
```

Cross Validation:

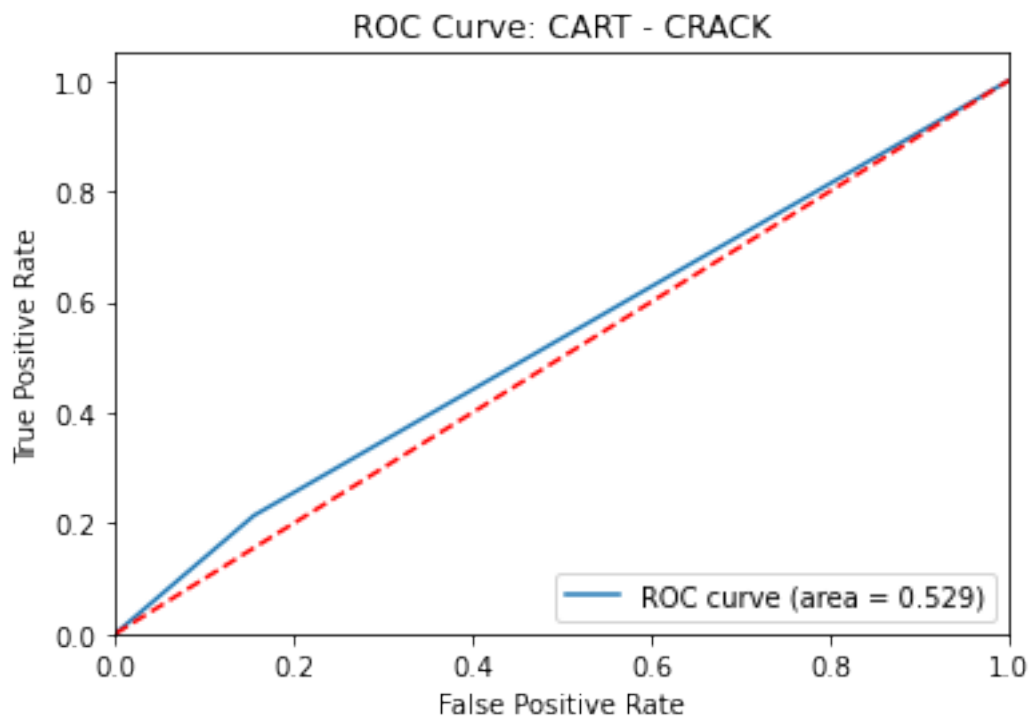
Mean Accuracy: 0.76233 +/- 0.02770

Mean Precision: 0.19724 +/- 0.02825

Mean F1 score: 0.20 +/- 0.07

Mean Recall: 0.22791 +/- 0.04743

Mean AUC Score: 0.53313 +/- 0.01950



--ECSTASY

Train-Test Split:

Accuracy: 0.650

Precision: 0.613

Recall: 0.592

F1 score: 0.602

Confusion matrix:

[[145 63]

[ 69 100]]

Cross Validation:

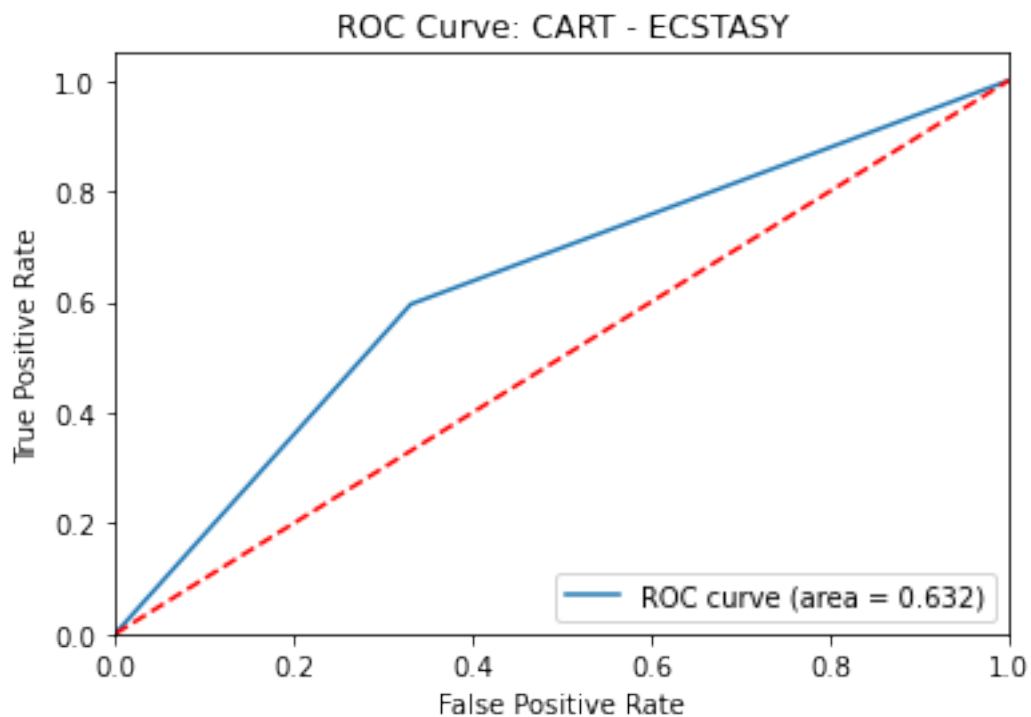
Mean Accuracy: 0.62706 +/- 0.02429

Mean Precision: 0.59618 +/- 0.02800

Mean F1 score: 0.60 +/- 0.04

Mean Recall: 0.58417 +/- 0.01843

Mean AUC Score: 0.62359 +/- 0.02186



--HEROIN

Train-Test Split:

Accuracy: 0.767

Precision: 0.273

Recall: 0.396

F1 score: 0.323

Confusion matrix:

```
[[268  56]
```

```
 [ 32  21]]
```

Cross Validation:

Mean Accuracy: 0.77878 +/- 0.04494

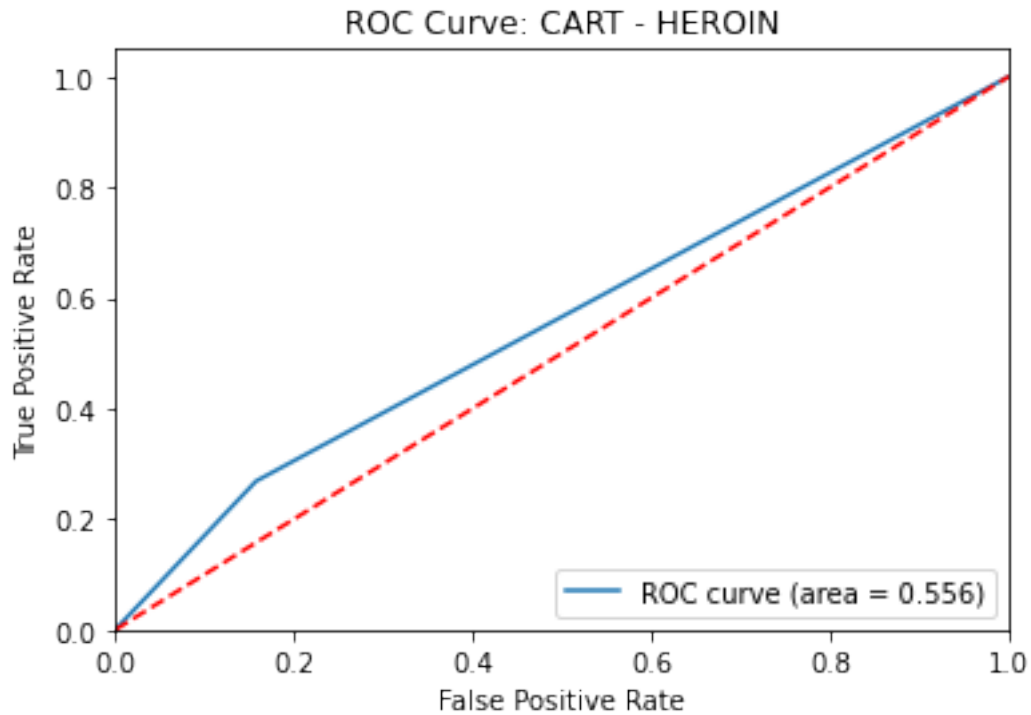
Mean Precision: 0.22319 +/- 0.05531

Mean F1 score: 0.31 +/- 0.02

Mean Recall: 0.26908 +/- 0.05641

Mean AUC Score: 0.54752 +/- 0.02606





--KETAMINE

Train-Test Split:

Accuracy: 0.714

Precision: 0.281

Recall: 0.362

F1 score: 0.316

Confusion matrix:

```
[[244  64]
```

```
 [ 44  25]]
```

Cross Validation:

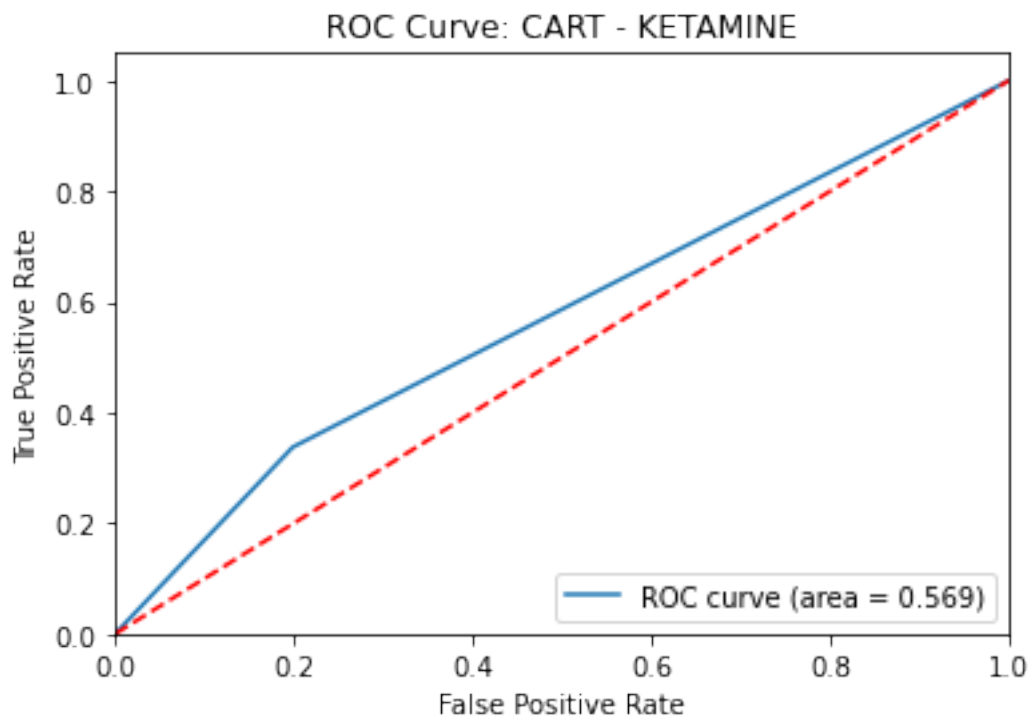
Mean Accuracy: 0.68488 +/- 0.04424

Mean Precision: 0.30796 +/- 0.04747

Mean F1 score: 0.27 +/- 0.03

Mean Recall: 0.33413 +/- 0.07238

Mean AUC Score: 0.56673 +/- 0.03135



--LEGALH

Train-Test Split:

Accuracy: 0.684

Precision: 0.642

Recall: 0.590

F1 score: 0.615

Confusion matrix:

```
[[163  53]
```

```
 [ 66  95]]
```

Cross Validation:

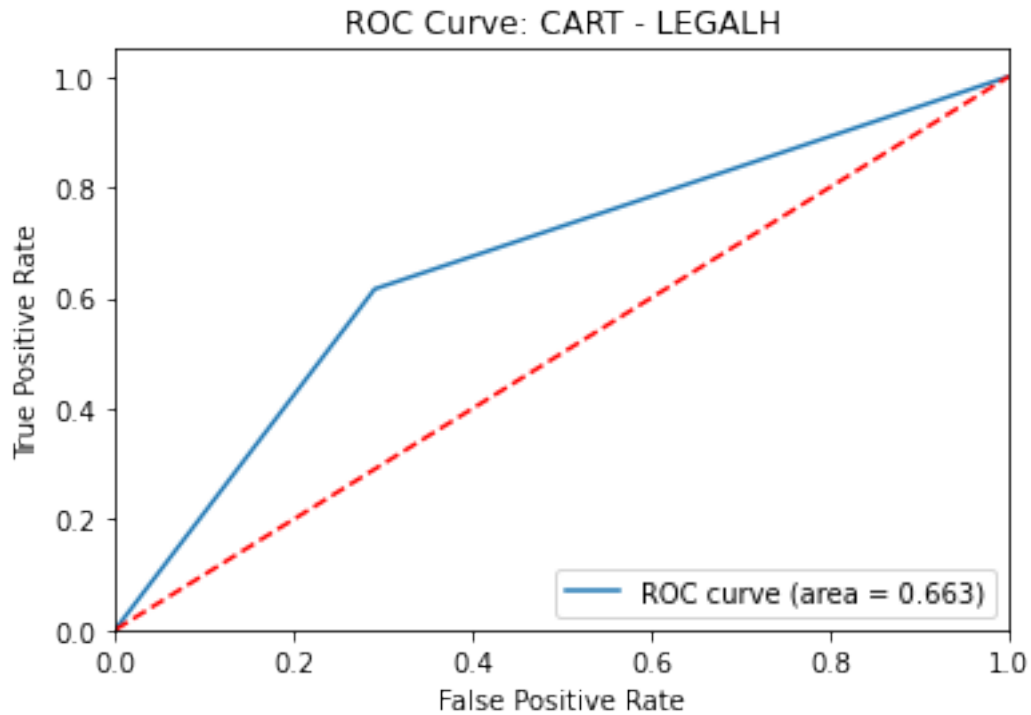
Mean Accuracy: 0.68170 +/- 0.03649

Mean Precision: 0.60752 +/- 0.01995

Mean F1 score: 0.61 +/- 0.06

Mean Recall: 0.62063 +/- 0.01166

Mean AUC Score: 0.66637 +/- 0.01395



--LSD

Train-Test Split:

Accuracy: 0.660

Precision: 0.601

Recall: 0.579

F1 score: 0.590

Confusion matrix:

```
[[157  61]
```

```
 [ 67  92]]
```

Cross Validation:

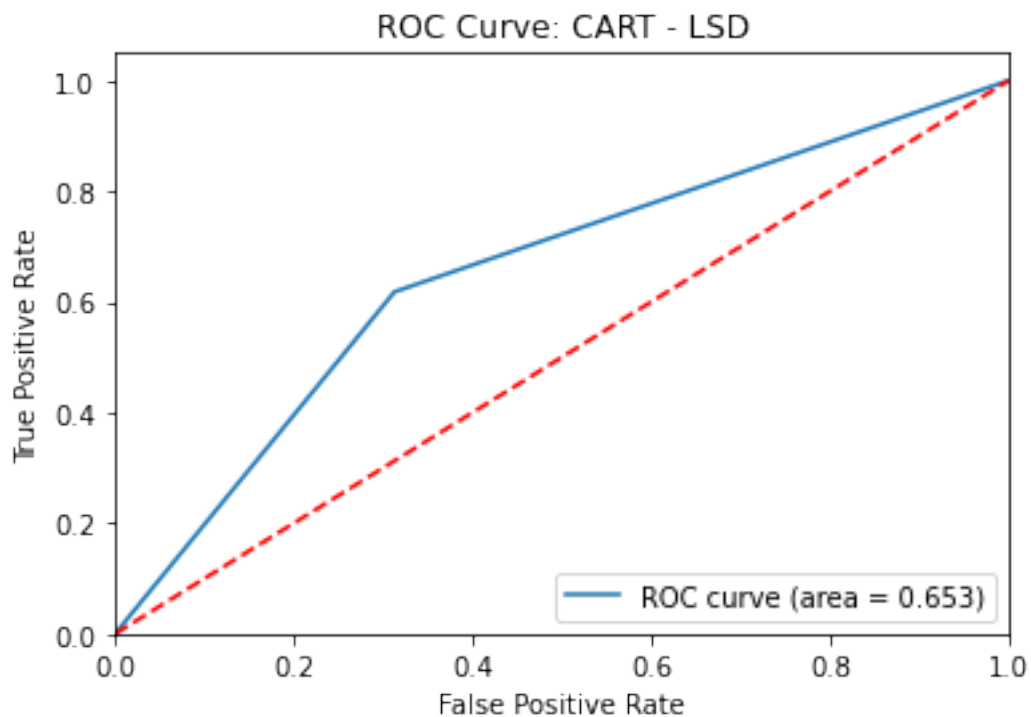
Mean Accuracy: 0.65836 +/- 0.03548

Mean Precision: 0.60328 +/- 0.03428

Mean F1 score: 0.61 +/- 0.04

Mean Recall: 0.61950 +/- 0.02281

Mean AUC Score: 0.64895 +/- 0.02881



--METH

Train-Test Split:

Accuracy: 0.700

Precision: 0.400

Recall: 0.430

F1 score: 0.415

Confusion matrix:

```
[[224  60]
```

```
 [ 53  40]]
```

Cross Validation:

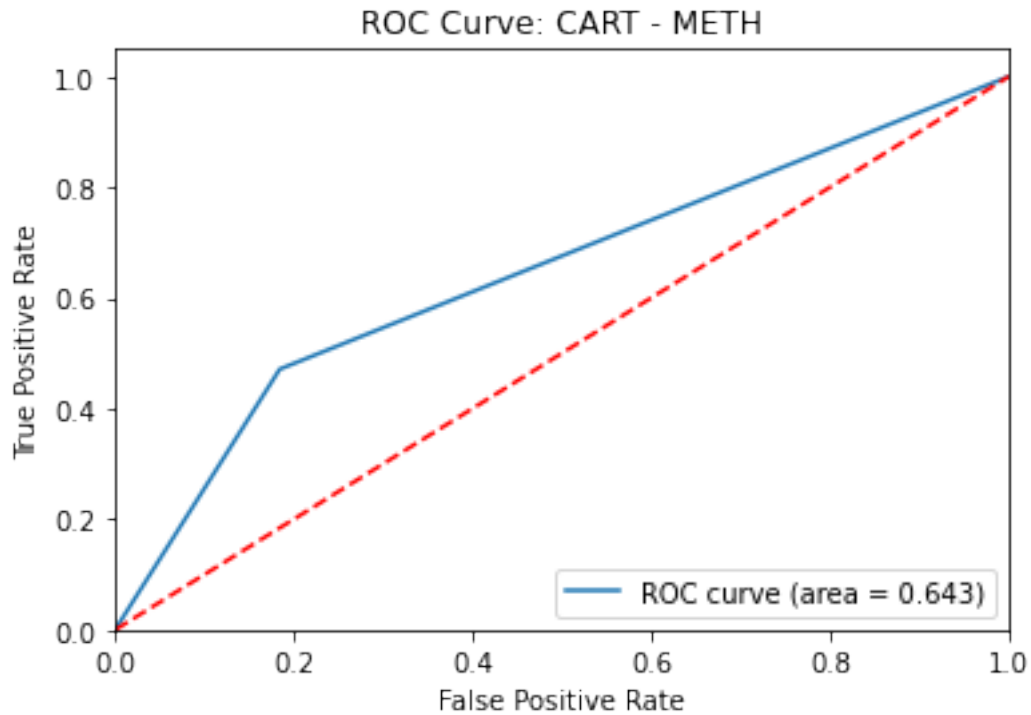
Mean Accuracy: 0.69231 +/- 0.03043

Mean Precision: 0.41941 +/- 0.05160

Mean F1 score: 0.38 +/- 0.05

Mean Recall: 0.45141 +/- 0.09783

Mean AUC Score: 0.63936 +/- 0.05084



--MUSHROOMS

Train-Test Split:

Accuracy: 0.639

Precision: 0.645

Recall: 0.597

F1 score: 0.620

Confusion matrix:

```
[[130  61]
```

```
 [ 75 111]]
```

Cross Validation:

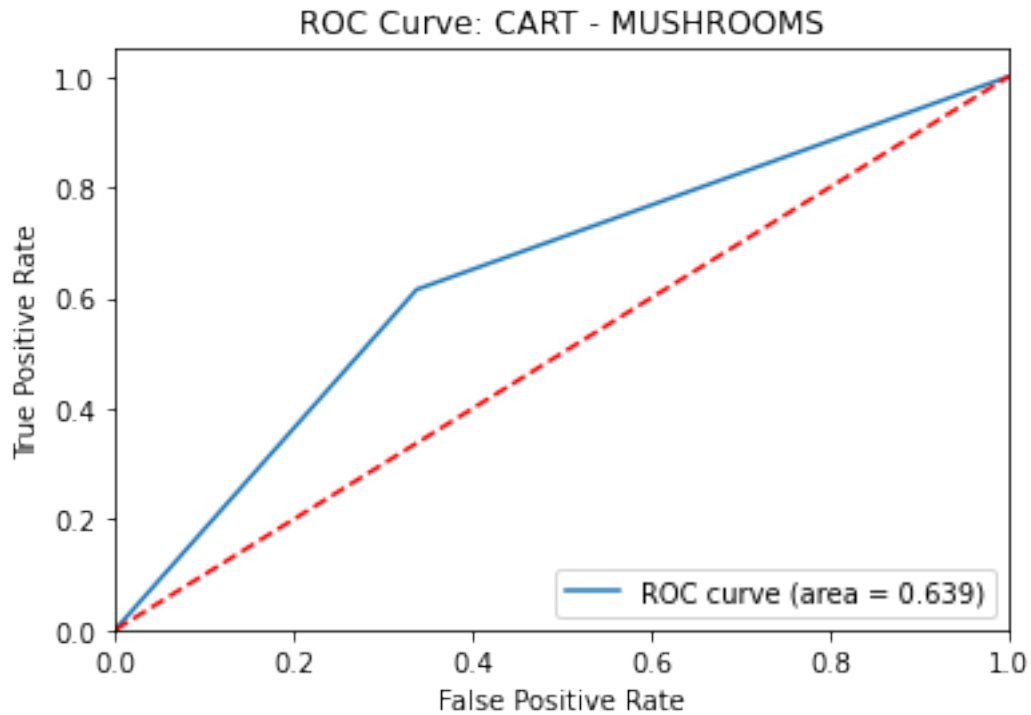
Mean Accuracy: 0.62122 +/- 0.03914

Mean Precision: 0.61875 +/- 0.02487

Mean F1 score: 0.62 +/- 0.05

Mean Recall: 0.60383 +/- 0.03071

Mean AUC Score: 0.64105 +/- 0.01316



--NICOTINE

Train-Test Split:

Accuracy: 0.647

Precision: 0.784

Recall: 0.738

F1 score: 0.760

Confusion matrix:

```
[[ 33  58]
```

```
 [ 75 211]]
```

Cross Validation:

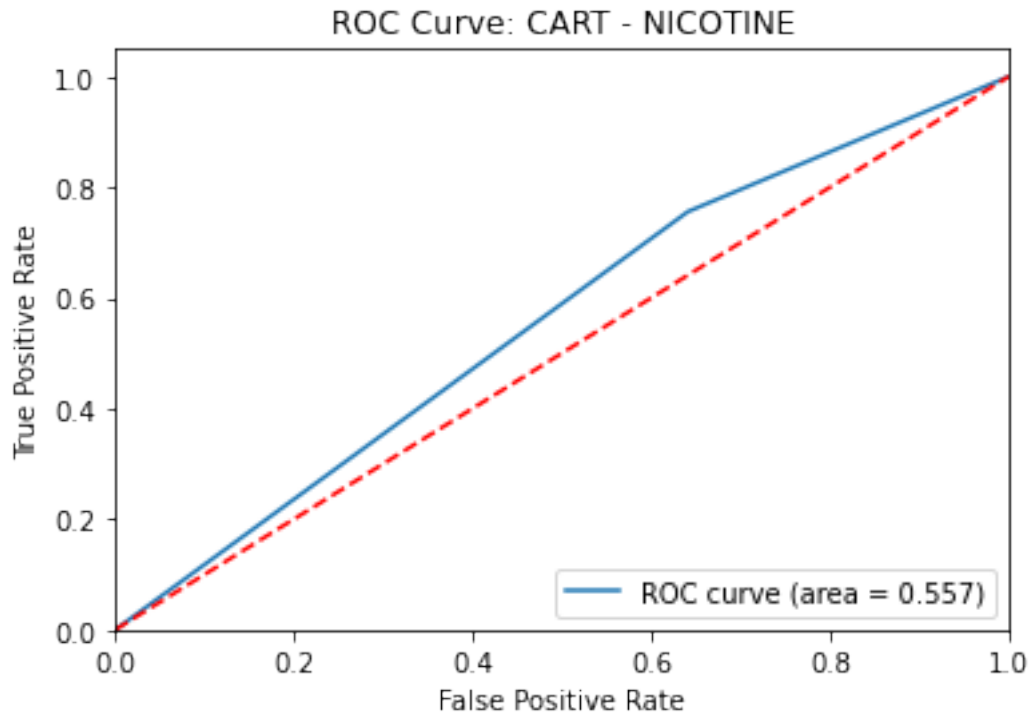
Mean Accuracy: 0.66472 +/- 0.03513

Mean Precision: 0.79827 +/- 0.01145

Mean F1 score: 0.78 +/- 0.03

Mean Recall: 0.76348 +/- 0.03239

Mean AUC Score: 0.55218 +/- 0.03680



## 12.4 Gaussian Naive Bayes (NB)

[13]:

```
df = pd.read_csv('data6.csv')
print("")

for col_index in range(9, len(df.columns)):

    columnName = df.columns[col_index]
    print()
    print("--" + (columnName))
    print()

    print('Train-Test Split:')

    X = df.iloc[:, 1:9]
    y = df[columnName]

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
↳ random_state=42)
```

```

# Create an instance of the model
model_1 = GaussianNB()

# Train the model on the training data
model_1.fit(X_train, y_train)

# Make predictions on the test data
y_pred1 = model_1.predict(X_test)

# Calculate evaluation metrics
accuracy = accuracy_score(y_test, y_pred1)
precision = precision_score(y_test, y_pred1)
recall = recall_score(y_test, y_pred1)
f1 = f1_score(y_test, y_pred1)
conf_matrix = confusion_matrix(y_test, y_pred1)

# Print the evaluation metrics
print(f"Accuracy: {accuracy:.3f}")
print(f"Precision: {precision:.3f}")
print(f"Recall: {recall:.3f}")
print(f"F1 score: {f1:.3f}")
print("Confusion matrix:")
print(conf_matrix)
print()

N_FOLDS = 5
# Perform cross-validation predictions
y_scores = cross_val_predict(model_1, X_train, y_train, cv=N_FOLDS,
↪method="predict_proba")

fpr, tpr, thresholds = roc_curve(y_train, y_scores[:, 1])
roc_auc = roc_auc_score(y_train, y_scores[:, 1])

print('Cross Validation:')
cv_accuracy = cross_val_score(model_1, X, y, cv=5, scoring='accuracy')
print('Mean Accuracy: %.5f +/- %.5f' % (np.mean(cv_accuracy), np.
↪std(cv_accuracy)))
precision_scores = cross_val_score(model_1, X_train, y_train, cv=N_FOLDS,
↪scoring="precision")

```



```

    print('Mean Precision: %.5f +/- %.5f' % (np.mean(precision_scores), np.
↪std(precision_scores)))
    recall_scores = cross_val_score(model_1, X_train, y_train, cv=N_FOLDS,
↪scoring="recall")
    cv_f1 = cross_val_score(model_1, X, y, cv=5, scoring='f1')
    print('Mean Recall: %.5f +/- %.5f' % (np.mean(recall_scores), np.
↪std(recall_scores)))
    print(f"Mean F1 score: {np.mean(cv_f1):.2f} +/- {np.std(cv_f1):.2f}")
    auc_scores = cross_val_score(model_1, X_train, y_train, cv=N_FOLDS,
↪scoring="roc_auc")
    print('Mean AUC Score: %.5f +/- %.5f' % (np.mean(auc_scores), np.
↪std(auc_scores)))
    print()
    print()

    # Plot ROC curve
    plt.figure()
    plt.plot(fpr, tpr, label='ROC curve (area = %0.3f)' % roc_auc)
    plt.plot([0, 1], [0, 1], 'r--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    # Model
    plt.title('ROC Curve: NB - ' + (columnName))
    plt.legend(loc="lower right")
    plt.show()

```

--AMPHET

Train-Test Split:

Accuracy: 0.650

Precision: 0.637

Recall: 0.630

F1 score: 0.633

Confusion matrix:

```
[[131  65]
```

```
 [ 67 114]]
```

Cross Validation:

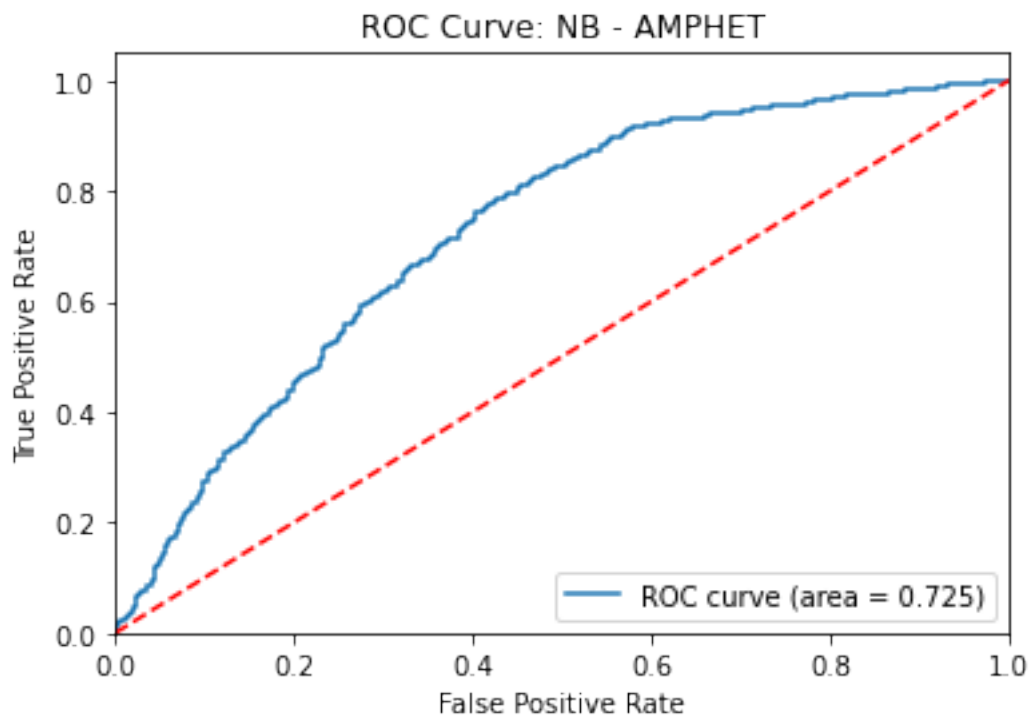
Mean Accuracy: 0.65199 +/- 0.04534

Mean Precision: 0.65248 +/- 0.00488

Mean Recall: 0.66209 +/- 0.03594

Mean F1 score: 0.62 +/- 0.12

Mean AUC Score: 0.72584 +/- 0.00353



--AMYL

Train-Test Split:

Accuracy: 0.700

Precision: 0.535

Recall: 0.198

F1 score: 0.289

Confusion matrix:

```
[[241  20]
```

```
 [ 93  23]]
```

Cross Validation:

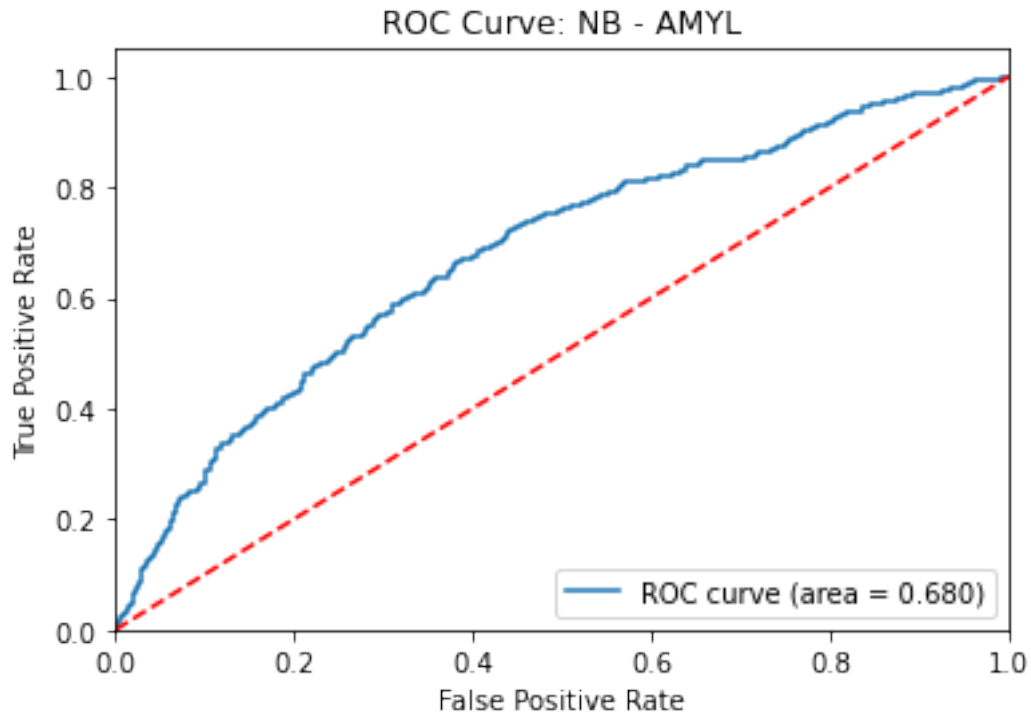
Mean Accuracy: 0.70875 +/- 0.01689

Mean Precision: 0.56852 +/- 0.08174

Mean Recall: 0.24780 +/- 0.04540

Mean F1 score: 0.34 +/- 0.05

Mean AUC Score: 0.68140 +/- 0.03051



--BENZOS

Train-Test Split:

Accuracy: 0.668

Precision: 0.643

Recall: 0.632

F1 score: 0.638

Confusion matrix:

```
[[142  61]
```

```
 [ 64 110]]
```

Cross Validation:

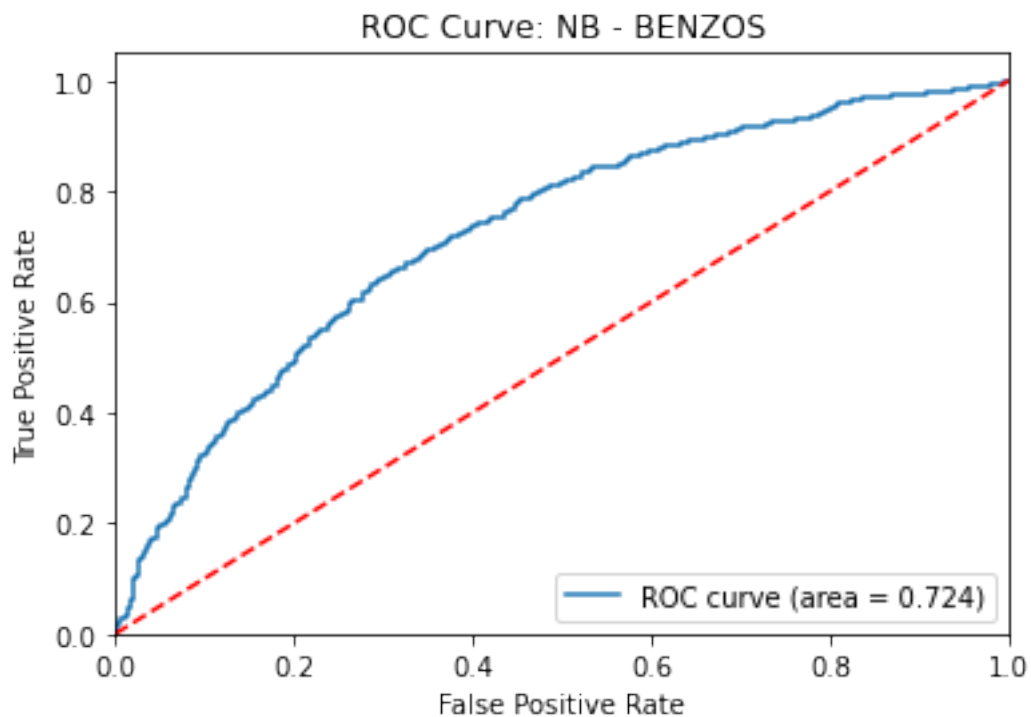
Mean Accuracy: 0.67003 +/- 0.03831

Mean Precision: 0.65448 +/- 0.02610

Mean Recall: 0.64690 +/- 0.04355

Mean F1 score: 0.63 +/- 0.10

Mean AUC Score: 0.72607 +/- 0.02962



--CANNABIS

Train-Test Split:

Accuracy: 0.748

Precision: 0.892

Recall: 0.766

F1 score: 0.824

Confusion matrix:

```
[[ 59  27]
```

```
 [ 68 223]]
```

Cross Validation:

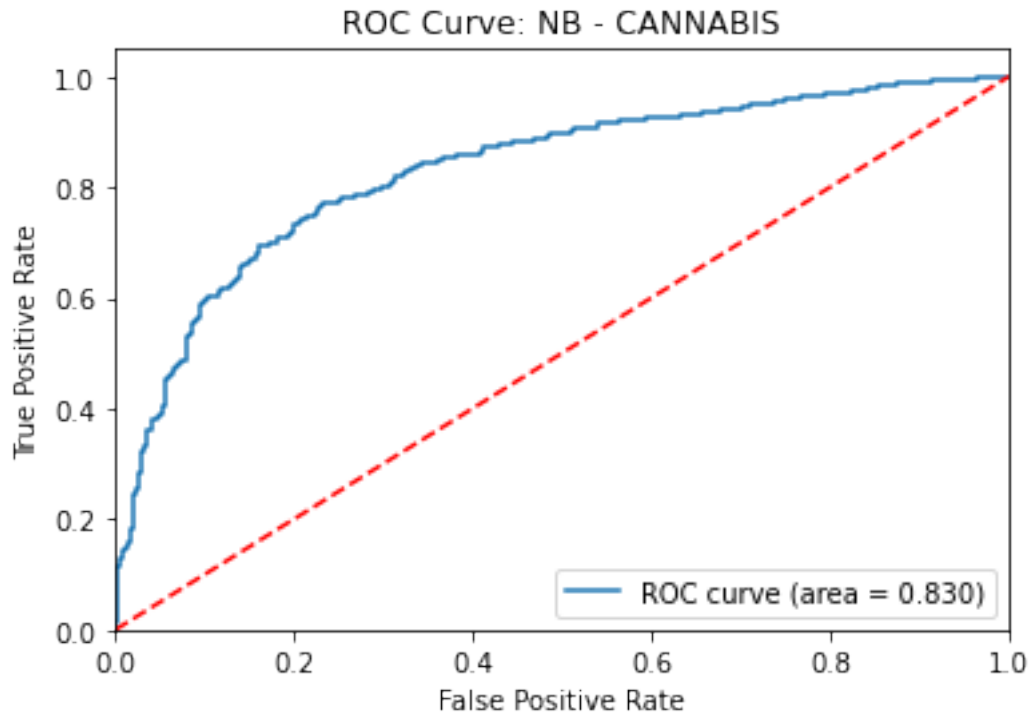
Mean Accuracy: 0.76180 +/- 0.11513

Mean Precision: 0.90909 +/- 0.01313

Mean Recall: 0.78996 +/- 0.03572

Mean F1 score: 0.82 +/- 0.11

Mean AUC Score: 0.83420 +/- 0.03386



--COKE

Train-Test Split:

Accuracy: 0.637

Precision: 0.548

Recall: 0.601

F1 score: 0.573

Confusion matrix:

```
[[148  76]
```

```
 [ 61  92]]
```

Cross Validation:

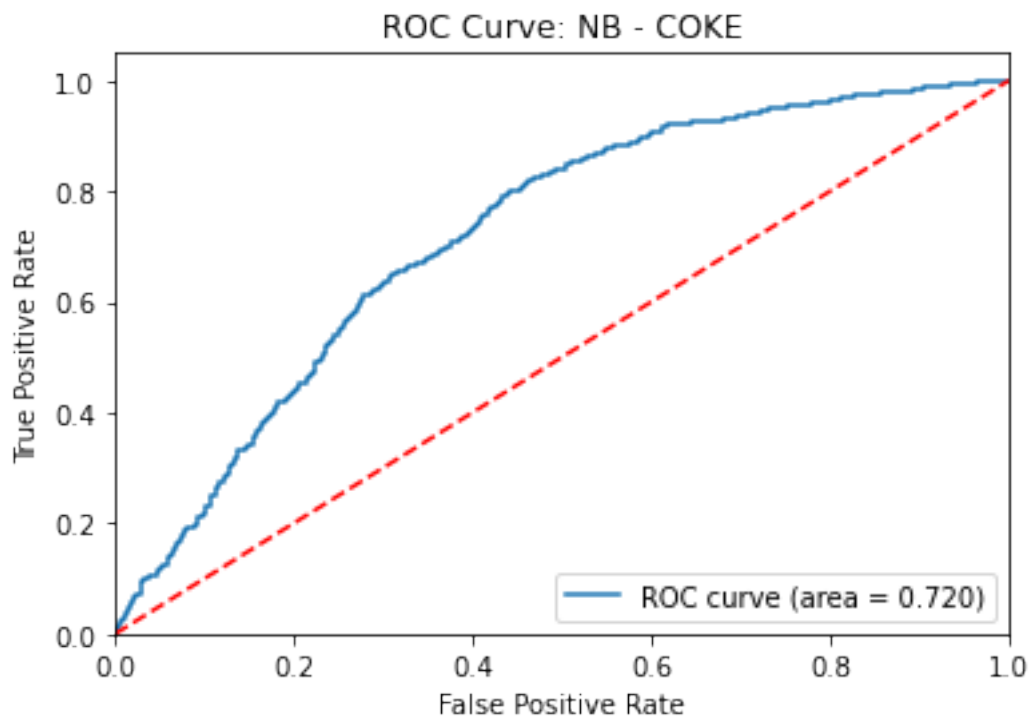
Mean Accuracy: 0.64934 +/- 0.06181

Mean Precision: 0.63709 +/- 0.00820

Mean Recall: 0.65412 +/- 0.05460

Mean F1 score: 0.60 +/- 0.08

Mean AUC Score: 0.72062 +/- 0.01542



--CRACK

Train-Test Split:

Accuracy: 0.801

Precision: 0.192

Recall: 0.233

F1 score: 0.211

Confusion matrix:

```
[[292  42]
```

```
 [ 33  10]]
```

Cross Validation:

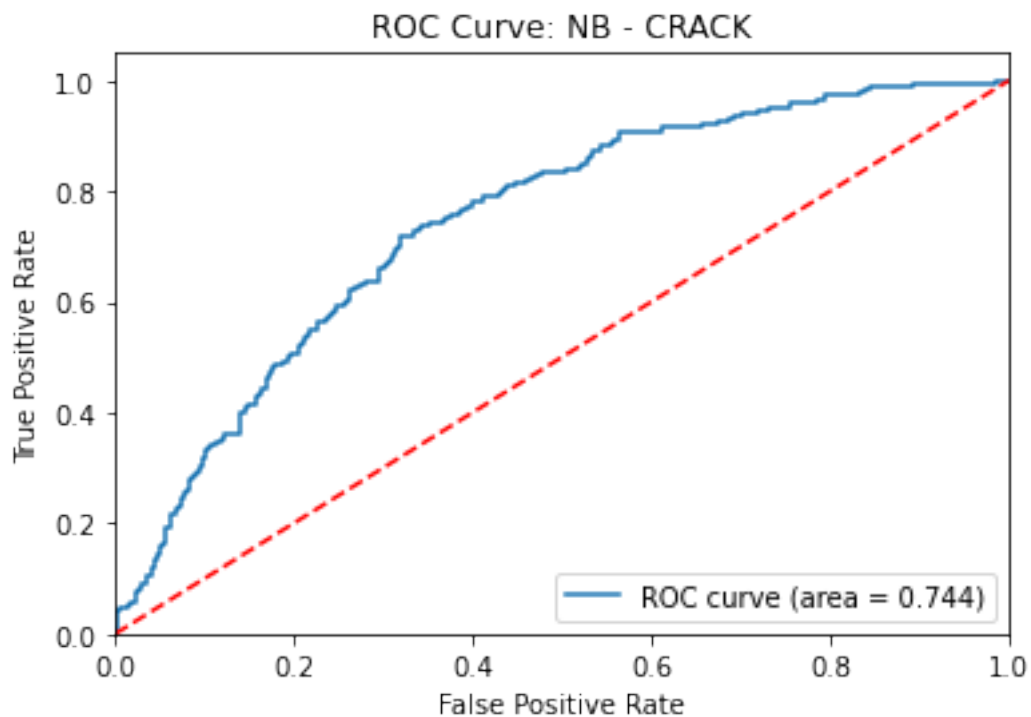
Mean Accuracy: 0.80690 +/- 0.06235

Mean Precision: 0.35007 +/- 0.07050

Mean Recall: 0.34419 +/- 0.05772

Mean F1 score: 0.29 +/- 0.09

Mean AUC Score: 0.74533 +/- 0.03916



--ECSTASY

Train-Test Split:

Accuracy: 0.716

Precision: 0.680

Recall: 0.692

F1 score: 0.686

Confusion matrix:

```
[[153  55]
```

```
 [ 52 117]]
```

Cross Validation:

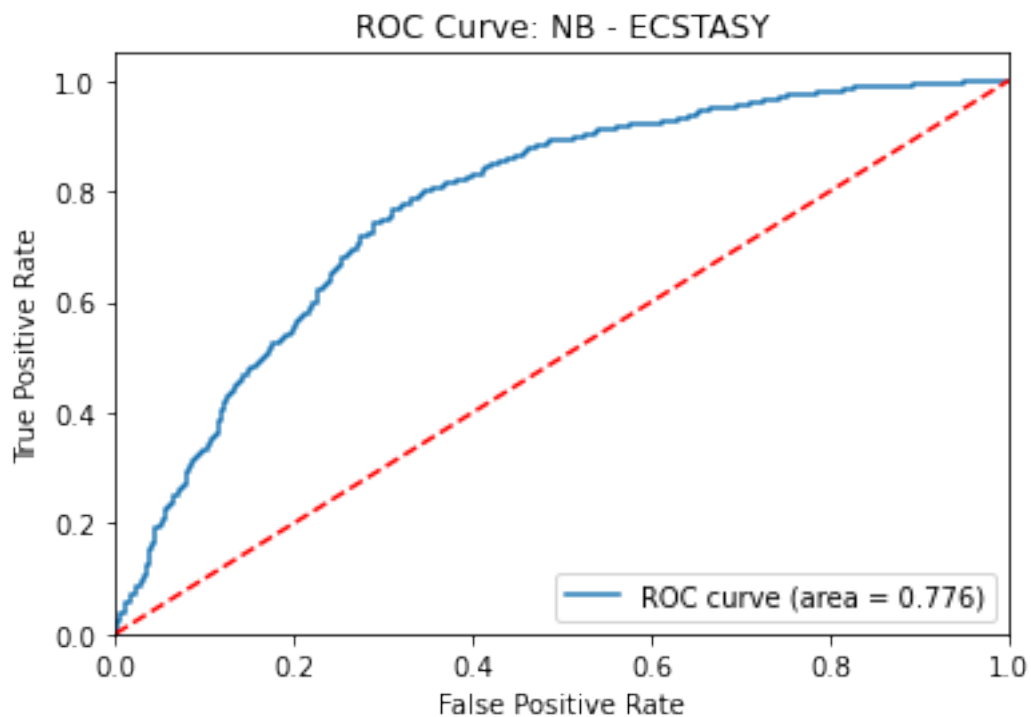
Mean Accuracy: 0.70292 +/- 0.06386

Mean Precision: 0.69079 +/- 0.03066

Mean Recall: 0.70216 +/- 0.03331

Mean F1 score: 0.66 +/- 0.12

Mean AUC Score: 0.77847 +/- 0.02382



--HEROIN

Train-Test Split:

Accuracy: 0.801

Precision: 0.317

Recall: 0.358

F1 score: 0.336

Confusion matrix:

```
[[283  41]
```

```
 [ 34  19]]
```

Cross Validation:

Mean Accuracy: 0.79310 +/- 0.06662

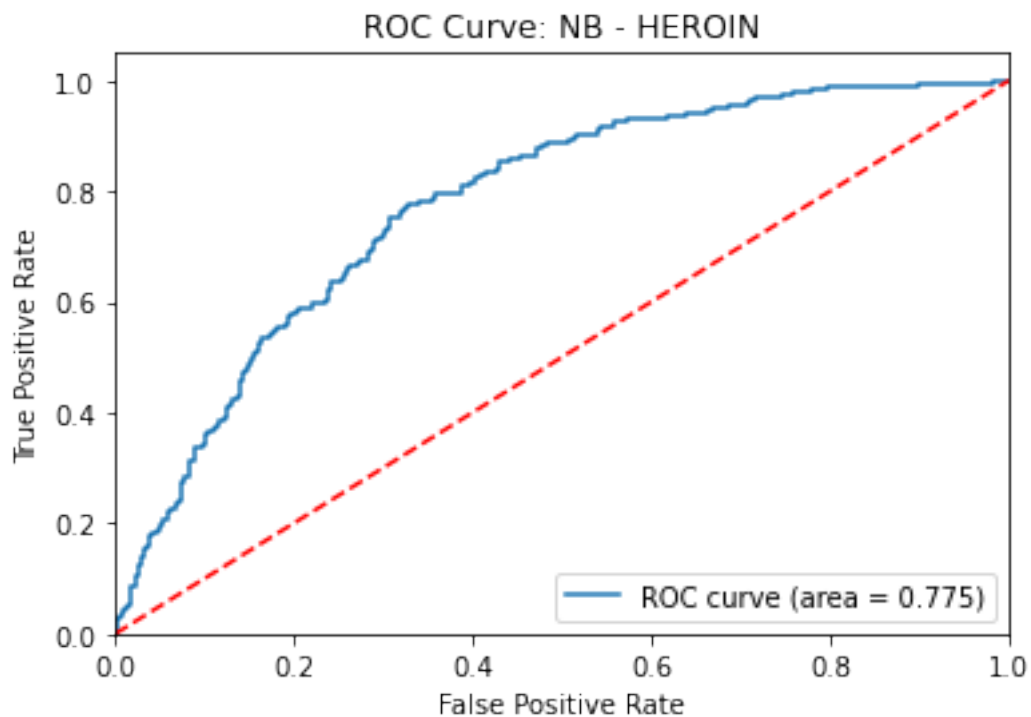
Mean Precision: 0.36411 +/- 0.08220

Mean Recall: 0.39729 +/- 0.08900

Mean F1 score: 0.35 +/- 0.09

Mean AUC Score: 0.77845 +/- 0.03723





--KETAMINE

Train-Test Split:

Accuracy: 0.751

Precision: 0.363

Recall: 0.478

F1 score: 0.413

Confusion matrix:

```
[[250  58]
```

```
 [ 36  33]]
```

Cross Validation:

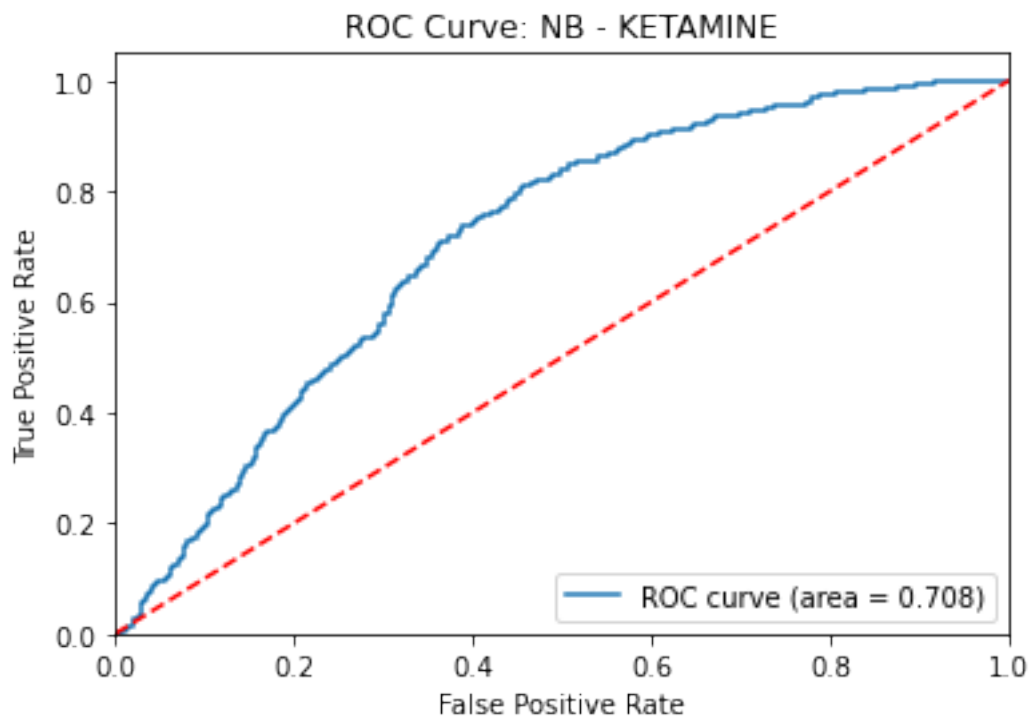
Mean Accuracy: 0.71353 +/- 0.08924

Mean Precision: 0.36281 +/- 0.01600

Mean Recall: 0.38042 +/- 0.01883

Mean F1 score: 0.34 +/- 0.14

Mean AUC Score: 0.70883 +/- 0.01886



--LEGALH

Train-Test Split:

Accuracy: 0.772

Precision: 0.730

Recall: 0.739

F1 score: 0.735

Confusion matrix:

```
[[172  44]
```

```
 [ 42 119]]
```

Cross Validation:

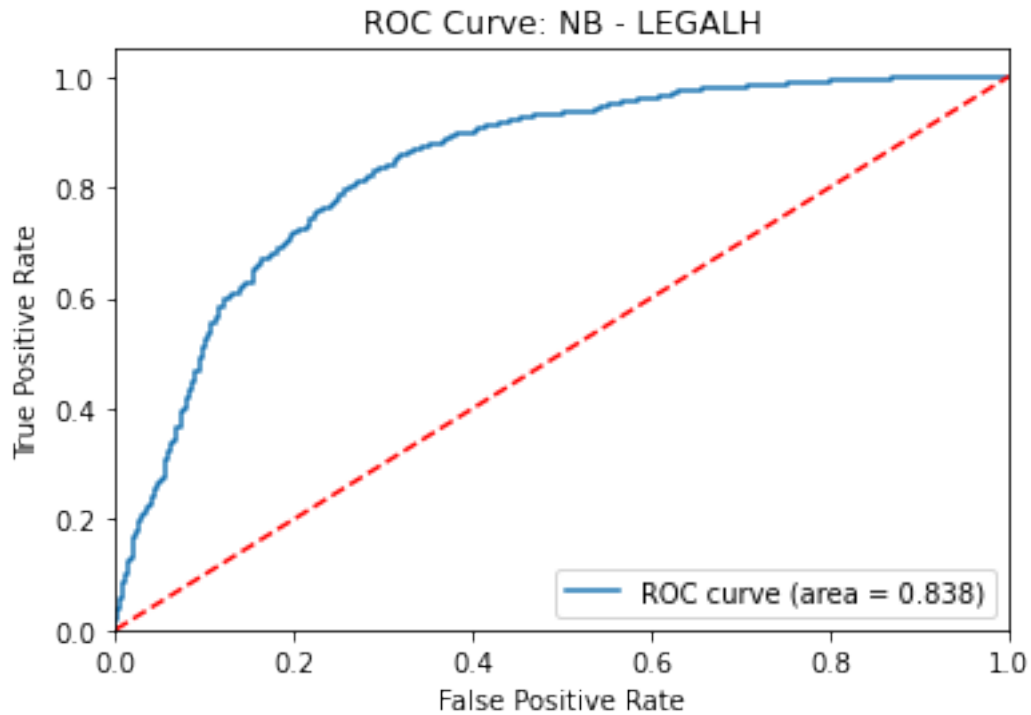
Mean Accuracy: 0.76021 +/- 0.06431

Mean Precision: 0.71209 +/- 0.03933

Mean Recall: 0.73016 +/- 0.03367

Mean F1 score: 0.70 +/- 0.12

Mean AUC Score: 0.83778 +/- 0.01678



--LSD

Train-Test Split:

Accuracy: 0.698

Precision: 0.633

Recall: 0.673

F1 score: 0.652

Confusion matrix:

```
[[156  62]
```

```
 [ 52 107]]
```

Cross Validation:

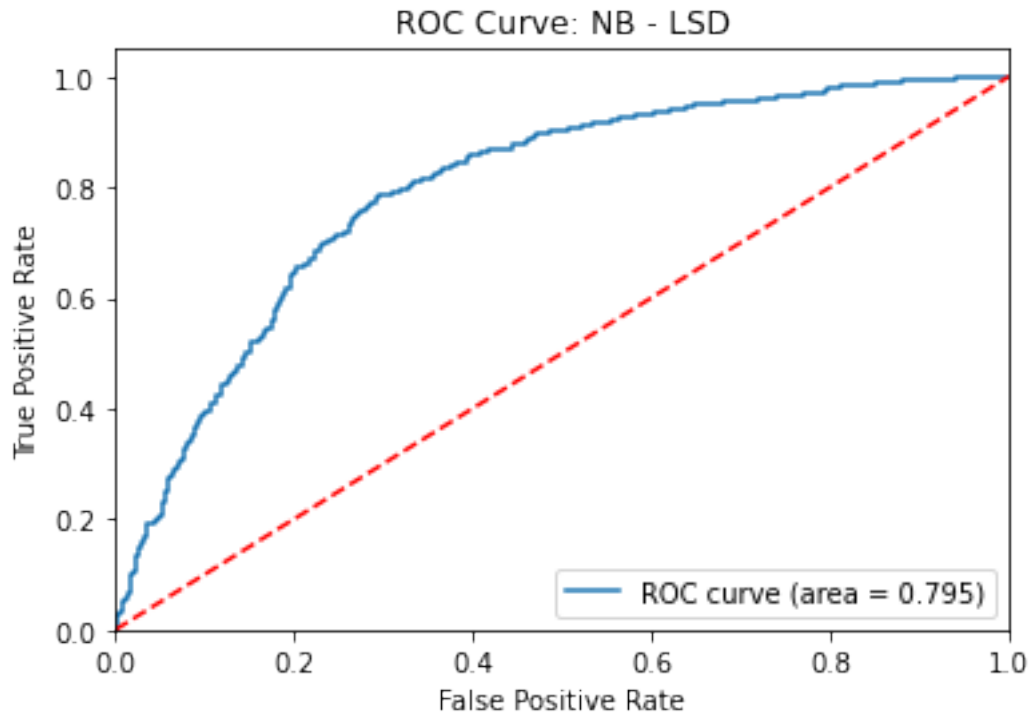
Mean Accuracy: 0.71300 +/- 0.05803

Mean Precision: 0.69042 +/- 0.02891

Mean Recall: 0.71389 +/- 0.01638

Mean F1 score: 0.65 +/- 0.15

Mean AUC Score: 0.79569 +/- 0.01093



--METH

Train-Test Split:

Accuracy: 0.732

Precision: 0.463

Recall: 0.538

F1 score: 0.498

Confusion matrix:

```
[[226  58]
```

```
 [ 43  50]]
```

Cross Validation:

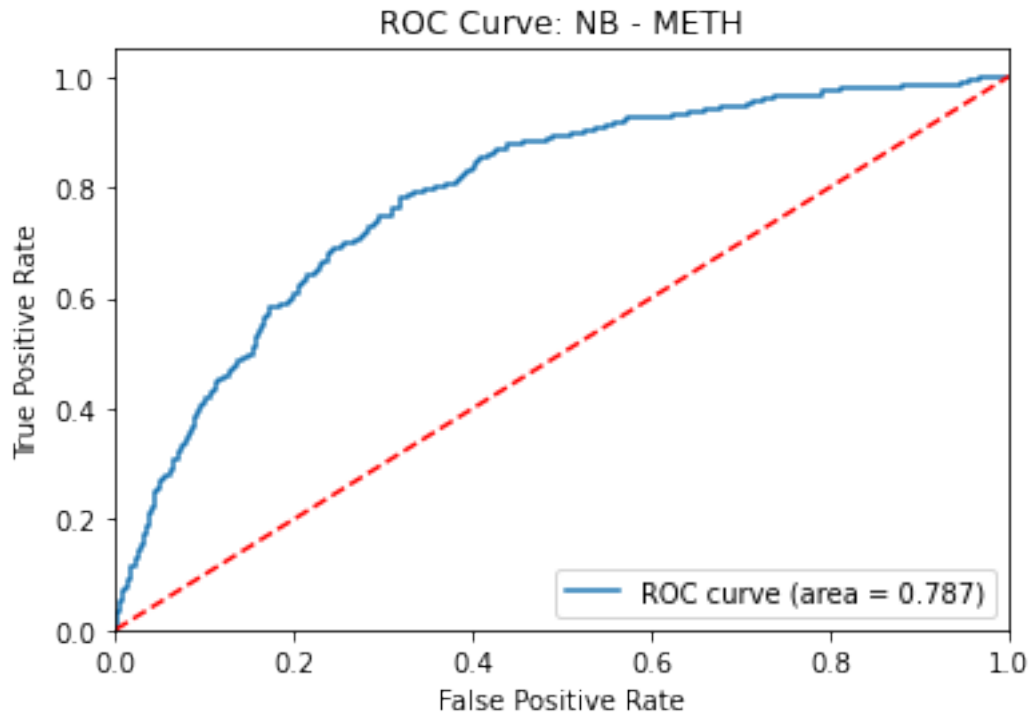
Mean Accuracy: 0.73846 +/- 0.08426

Mean Precision: 0.49847 +/- 0.02931

Mean Recall: 0.58957 +/- 0.04091

Mean F1 score: 0.50 +/- 0.14

Mean AUC Score: 0.78775 +/- 0.00857



--MUSHROOMS

Train-Test Split:

Accuracy: 0.692

Precision: 0.694

Recall: 0.672

F1 score: 0.683

Confusion matrix:

```
[[136  55]
```

```
 [ 61 125]]
```

Cross Validation:

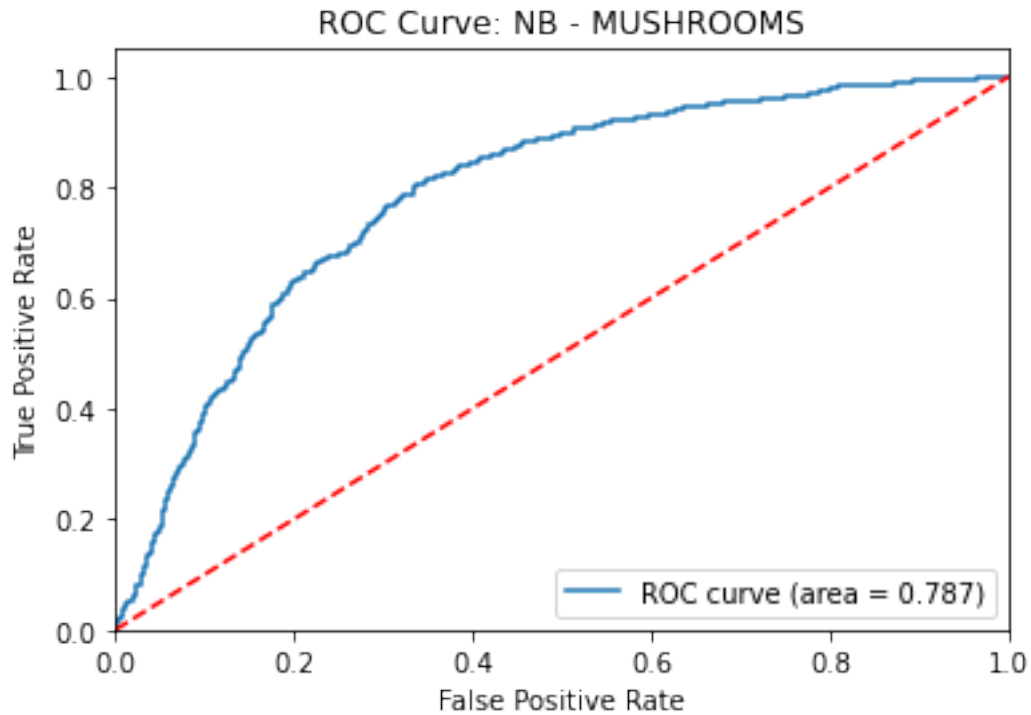
Mean Accuracy: 0.71141 +/- 0.06563

Mean Precision: 0.70042 +/- 0.02651

Mean Recall: 0.70155 +/- 0.03261

Mean F1 score: 0.68 +/- 0.13

Mean AUC Score: 0.78819 +/- 0.02532



--NICOTINE

Train-Test Split:

Accuracy: 0.714

Precision: 0.818

Recall: 0.801

F1 score: 0.809

Confusion matrix:

```
[[ 40  51]
```

```
 [ 57 229]]
```

Cross Validation:

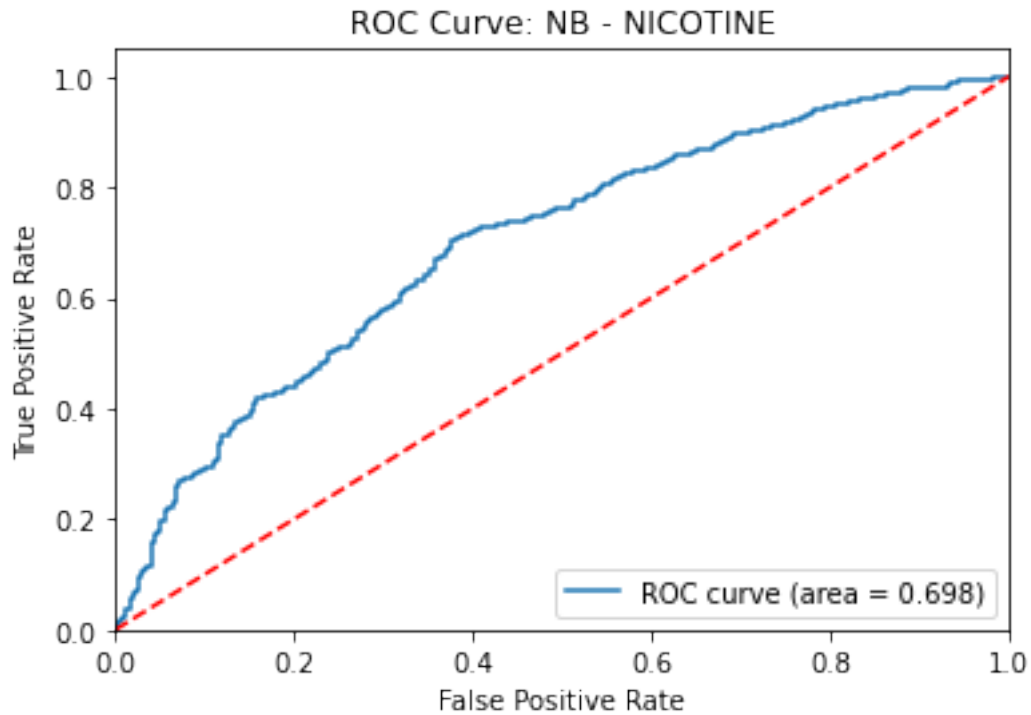
Mean Accuracy: 0.71406 +/- 0.08659

Mean Precision: 0.83490 +/- 0.00712

Mean Recall: 0.81978 +/- 0.02881

Mean F1 score: 0.80 +/- 0.09

Mean AUC Score: 0.69908 +/- 0.02495



## 12.5 Random Forest (RF)

[15]:

```
df = pd.read_csv('data6.csv')
print("")

for col_index in range(9, len(df.columns)):

    columnName = df.columns[col_index]
    print()
    print("--" + (columnName))
    print()

    print('Train-Test Split:')

    X = df.iloc[:, 1:9]
    y = df[columnName]

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
↳ random_state=42)
```

```

# Create an instance of the model
model_1 = RandomForestClassifier()

# Train the model on the training data
model_1.fit(X_train, y_train)

# Make predictions on the test data
y_pred1 = model_1.predict(X_test)

# Calculate evaluation metrics
accuracy = accuracy_score(y_test, y_pred1)
precision = precision_score(y_test, y_pred1)
recall = recall_score(y_test, y_pred1)
f1 = f1_score(y_test, y_pred1)
conf_matrix = confusion_matrix(y_test, y_pred1)

# Print the evaluation metrics
print(f"Accuracy: {accuracy:.3f}")
print(f"Precision: {precision:.3f}")
print(f"Recall: {recall:.3f}")
print(f"F1 score: {f1:.3f}")
print("Confusion matrix:")
print(conf_matrix)
print()

N_FOLDS = 5
# Perform cross-validation predictions
y_scores = cross_val_predict(model_1, X_train, y_train, cv=N_FOLDS,
↪method="predict_proba")

fpr, tpr, thresholds = roc_curve(y_train, y_scores[:, 1])
roc_auc = roc_auc_score(y_train, y_scores[:, 1])

print('Cross Validation:')
cv_accuracy = cross_val_score(model_1, X, y, cv=5, scoring='accuracy')
print('Mean Accuracy: %.5f +/- %.5f' % (np.mean(cv_accuracy), np.
↪std(cv_accuracy)))
precision_scores = cross_val_score(model_1, X_train, y_train, cv=N_FOLDS,
↪scoring="precision")
print('Mean Precision: %.5f +/- %.5f' % (np.mean(precision_scores), np.
↪std(precision_scores)))

```



```

    recall_scores = cross_val_score(model_1, X_train, y_train, cv=N_FOLDS,
↪scoring="recall")
    cv_f1 = cross_val_score(model_1, X, y, cv=5, scoring='f1')
    print('Mean Recall: %.5f +/- %.5f' % (np.mean(recall_scores), np.
↪std(recall_scores)))
    print(f"Mean F1 score: {np.mean(cv_f1):.2f} +/- {np.std(cv_f1):.2f}")
    auc_scores = cross_val_score(model_1, X_train, y_train, cv=N_FOLDS,
↪scoring="roc_auc")
    print('Mean AUC Score: %.5f +/- %.5f' % (np.mean(auc_scores), np.
↪std(auc_scores)))
    print()
    print()

    # Plot ROC curve
    plt.figure()
    plt.plot(fpr, tpr, label='ROC curve (area = %0.3f)' % roc_auc)
    plt.plot([0, 1], [0, 1], 'r--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    # Model
    plt.title('ROC Curve: RF - ' + (columnName))
    plt.legend(loc="lower right")
    plt.show()

```

--AMPHET

Train-Test Split:

Accuracy: 0.639

Precision: 0.618

Recall: 0.652

F1 score: 0.634

Confusion matrix:

```
[[123  73]
```

```
 [ 63 118]]
```

Cross Validation:

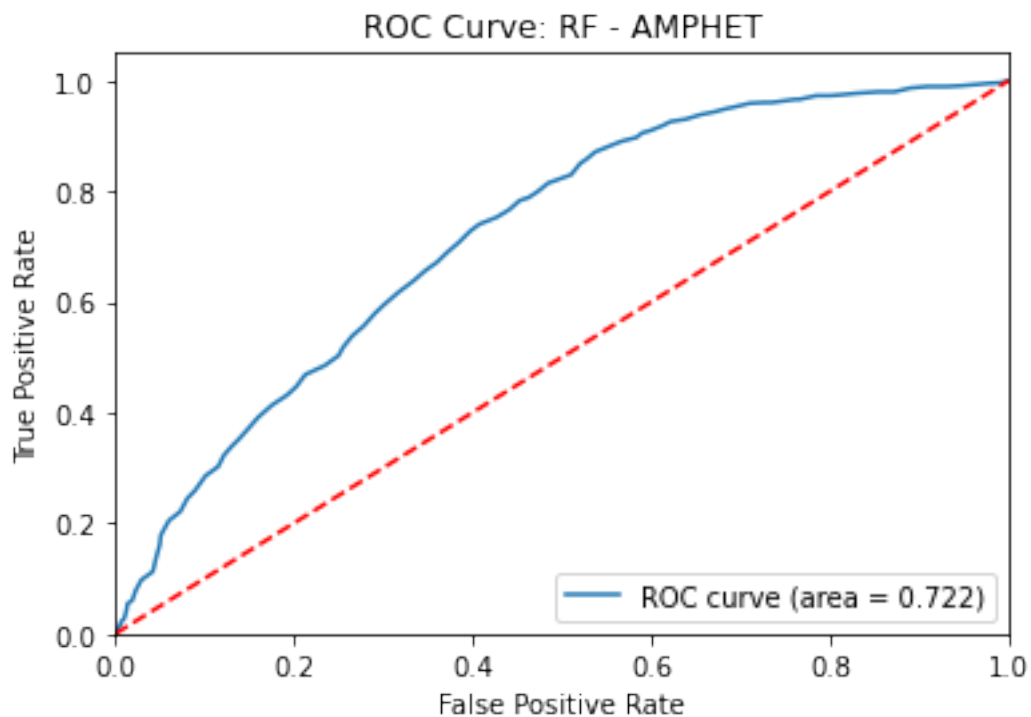
Mean Accuracy: 0.64509 +/- 0.03911

Mean Precision: 0.64145 +/- 0.01543

Mean Recall: 0.65932 +/- 0.03910

Mean F1 score: 0.63 +/- 0.07

Mean AUC Score: 0.71725 +/- 0.01819



--AMYL

Train-Test Split:

Accuracy: 0.716

Precision: 0.582

Recall: 0.276

F1 score: 0.374

Confusion matrix:

```
[[238  23]
```

```
 [ 84 32]]
```

Cross Validation:

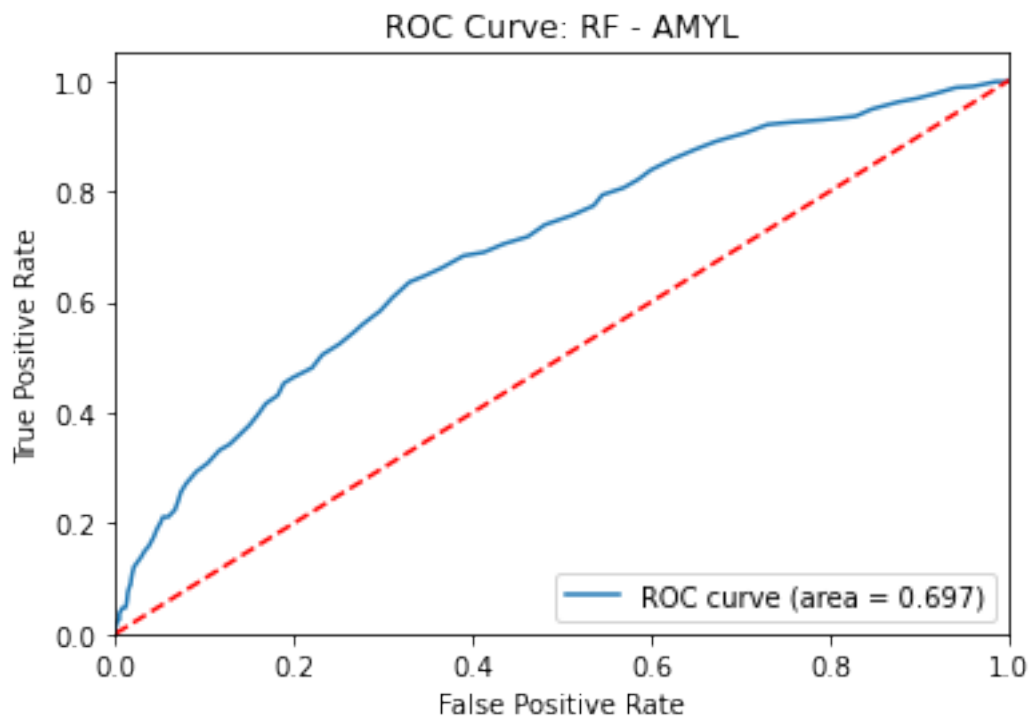
Mean Accuracy: 0.71671 +/- 0.01779

Mean Precision: 0.56714 +/- 0.03781

Mean Recall: 0.29088 +/- 0.01699

Mean F1 score: 0.40 +/- 0.05

Mean AUC Score: 0.69835 +/- 0.04280



--BENZOS

Train-Test Split:

Accuracy: 0.676

Precision: 0.659

Recall: 0.621

F1 score: 0.639

Confusion matrix:

```
[[147  56]
```

```
 [ 66 108]]
```

Cross Validation:

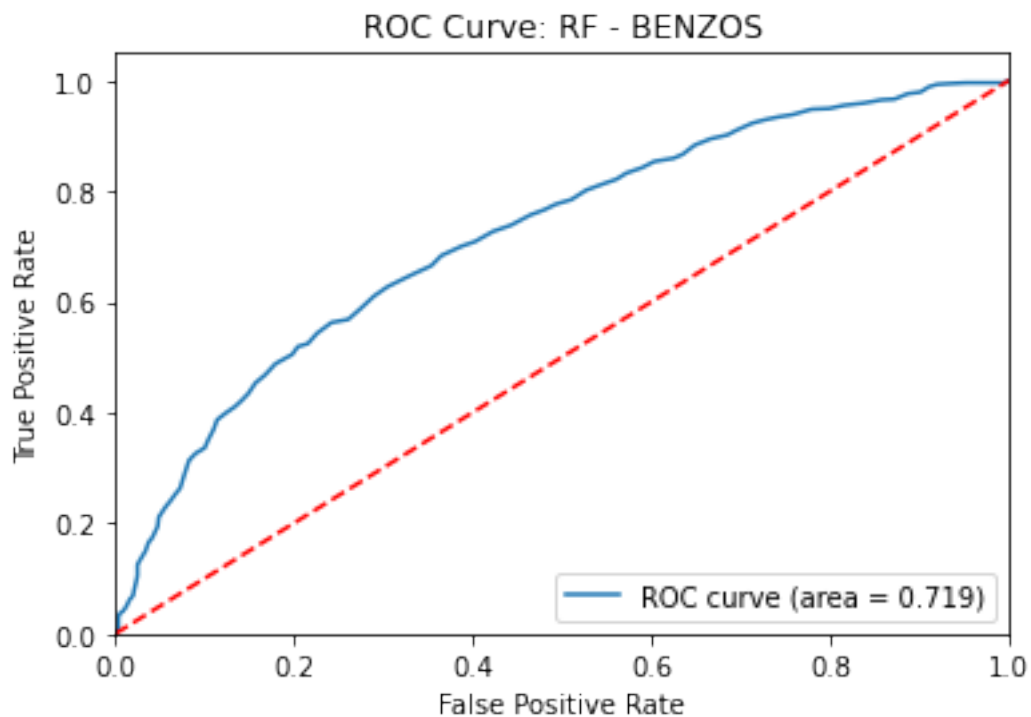
Mean Accuracy: 0.65570 +/- 0.02530

Mean Precision: 0.64380 +/- 0.01322

Mean Recall: 0.60192 +/- 0.03252

Mean F1 score: 0.61 +/- 0.06

Mean AUC Score: 0.72303 +/- 0.02961



--CANNABIS

Train-Test Split:

Accuracy: 0.793

Precision: 0.838

Recall: 0.907

F1 score: 0.871

Confusion matrix:

```
[[ 35  51]
```

```
 [ 27 264]]
```

Cross Validation:

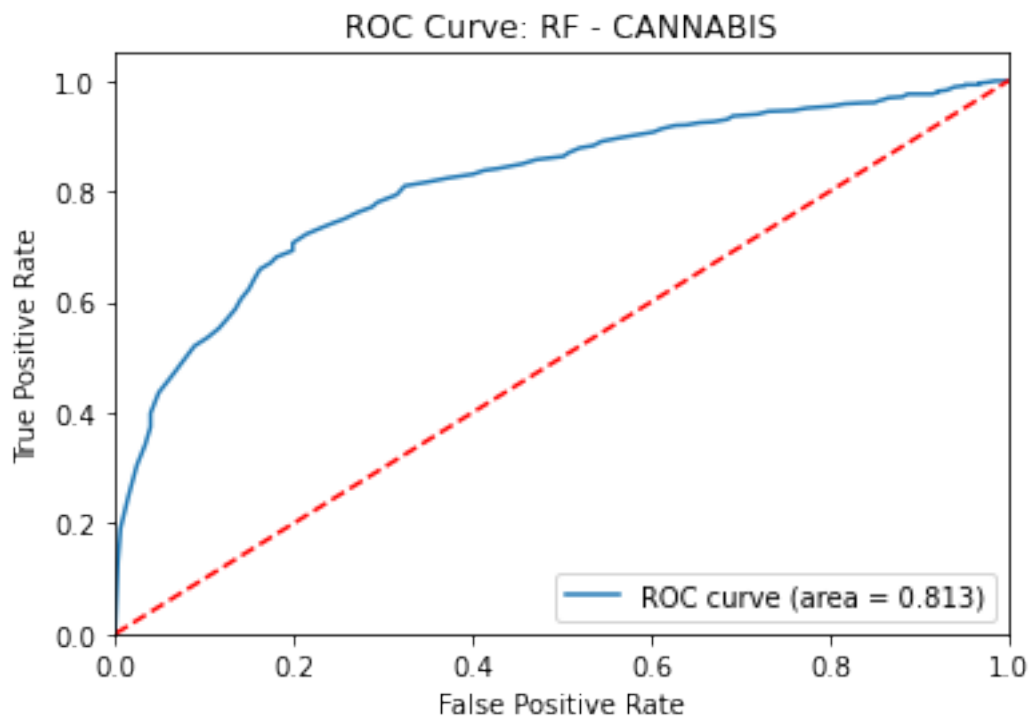
Mean Accuracy: 0.78992 +/- 0.05000

Mean Precision: 0.84007 +/- 0.01019

Mean Recall: 0.91615 +/- 0.01946

Mean F1 score: 0.87 +/- 0.03

Mean AUC Score: 0.81501 +/- 0.02196



--COKE

Train-Test Split:

Accuracy: 0.639

Precision: 0.556

Recall: 0.556

F1 score: 0.556

Confusion matrix:

```
[[156 68]
```

```
 [ 68 85]]
```

Cross Validation:

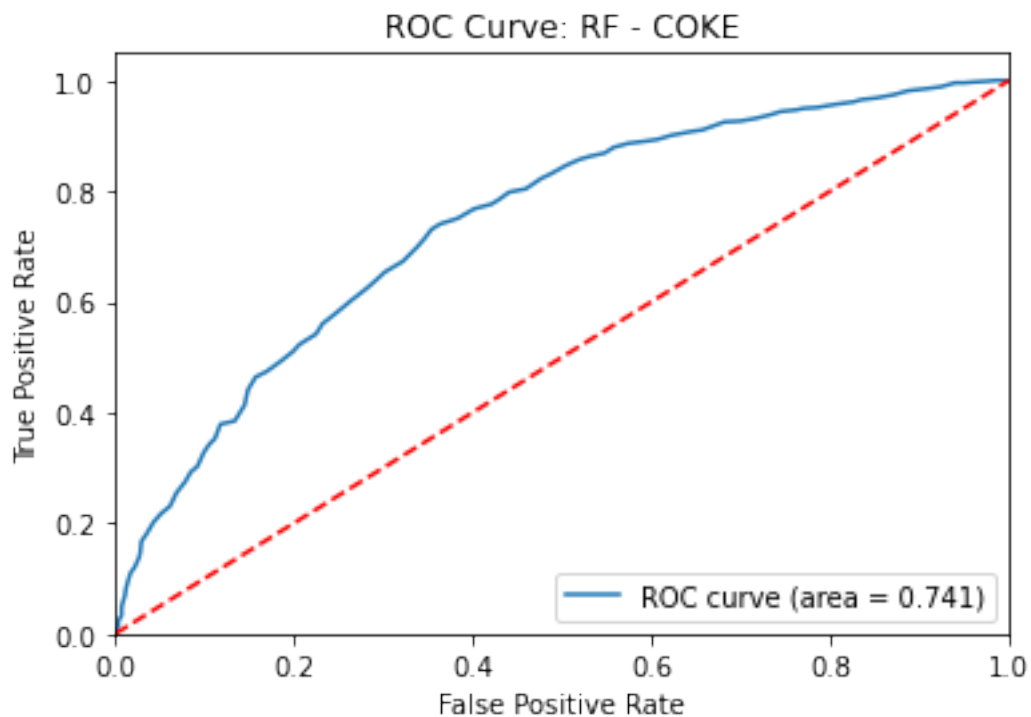
Mean Accuracy: 0.66578 +/- 0.04174

Mean Precision: 0.65603 +/- 0.01452

Mean Recall: 0.64264 +/- 0.02942

Mean F1 score: 0.61 +/- 0.05

Mean AUC Score: 0.74185 +/- 0.01255



--CRACK

Train-Test Split:

Accuracy: 0.881

Precision: 0.250

Recall: 0.023

F1 score: 0.043

Confusion matrix:

```
[[331  3]
```

```
 [ 42  1]]
```

Cross Validation:

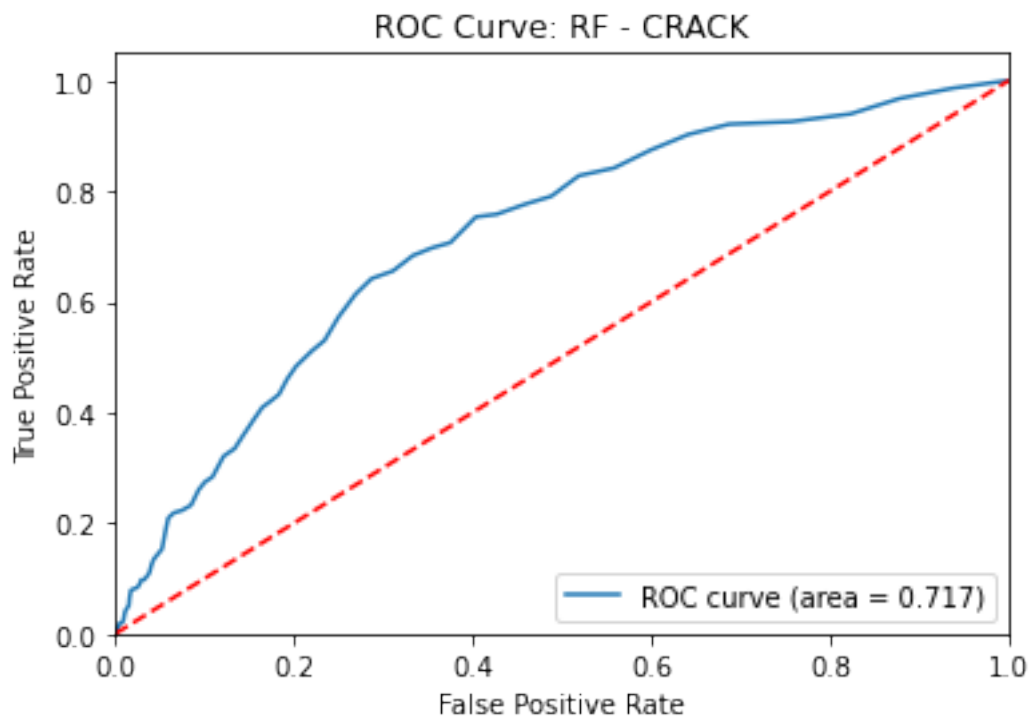
Mean Accuracy: 0.85782 +/- 0.00571

Mean Precision: 0.36905 +/- 0.08650

Mean Recall: 0.08837 +/- 0.03418

Mean F1 score: 0.11 +/- 0.07

Mean AUC Score: 0.71985 +/- 0.02742



--ECSTASY

Train-Test Split:

Accuracy: 0.700

Precision: 0.659

Recall: 0.686

F1 score: 0.672

Confusion matrix:

```
[[148  60]
```

```
 [ 53 116]]
```

Cross Validation:

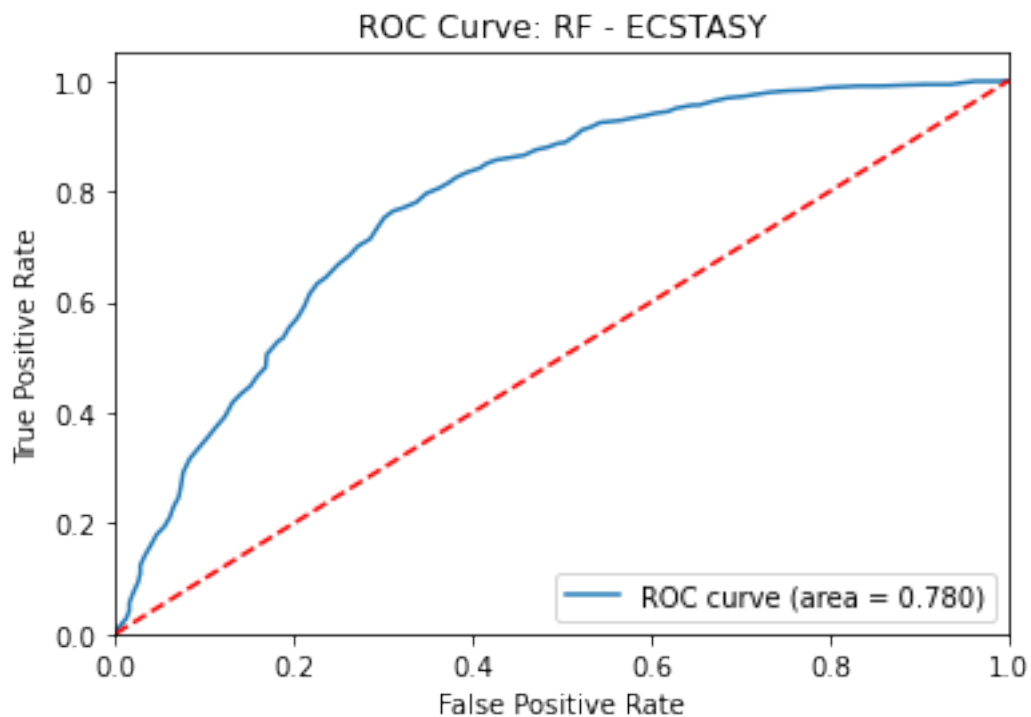
Mean Accuracy: 0.70875 +/- 0.04090

Mean Precision: 0.68724 +/- 0.02831

Mean Recall: 0.69640 +/- 0.03708

Mean F1 score: 0.68 +/- 0.09

Mean AUC Score: 0.78304 +/- 0.01029



--HEROIN

Train-Test Split:

Accuracy: 0.867

Precision: 0.636

Recall: 0.132

F1 score: 0.219

Confusion matrix:

```
[[320  4]
```

```
 [ 46  7]]
```

Cross Validation:

Mean Accuracy: 0.84297 +/- 0.01630

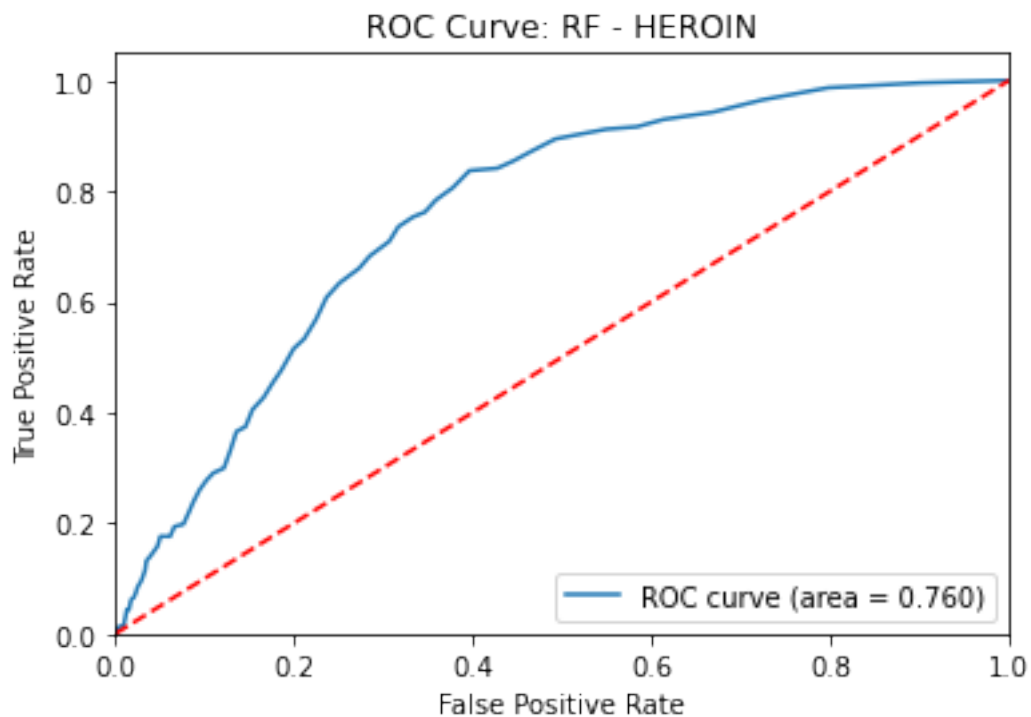
Mean Precision: 0.41576 +/- 0.13832

Mean Recall: 0.07507 +/- 0.03015

Mean F1 score: 0.15 +/- 0.05

Mean AUC Score: 0.75996 +/- 0.04153





--KETAMINE

Train-Test Split:

Accuracy: 0.780

Precision: 0.250

Recall: 0.101

F1 score: 0.144

Confusion matrix:

```
[[287  21]
```

```
 [ 62   7]]
```

Cross Validation:

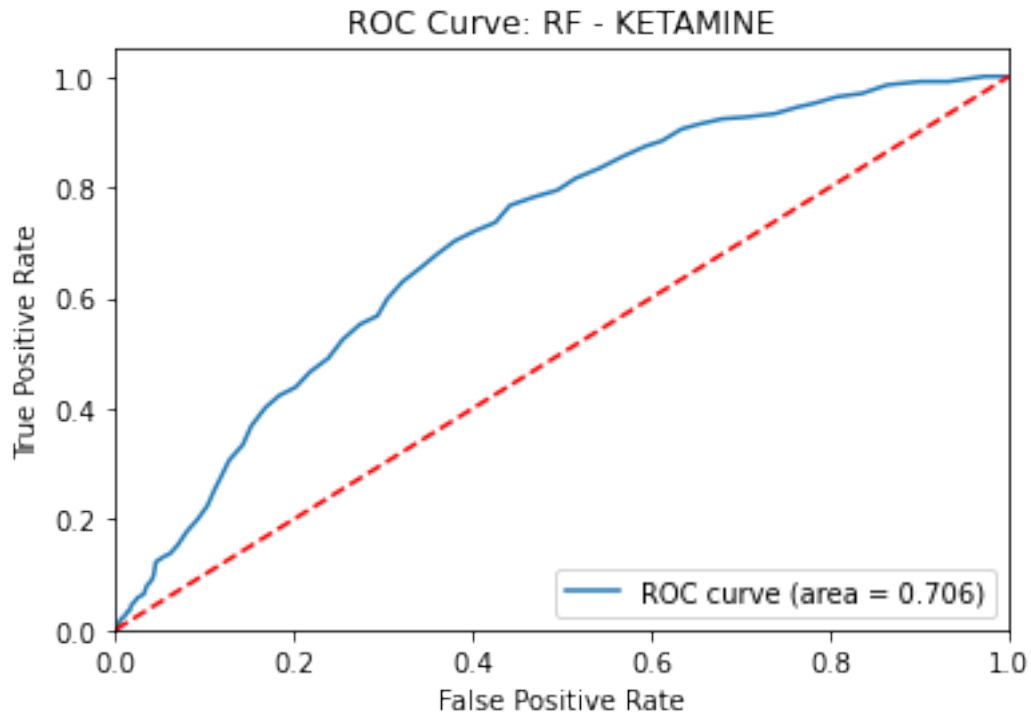
Mean Accuracy: 0.76605 +/- 0.01871

Mean Precision: 0.39928 +/- 0.20620

Mean Recall: 0.09497 +/- 0.03644

Mean F1 score: 0.12 +/- 0.04

Mean AUC Score: 0.70321 +/- 0.02878



--LEGALH

Train-Test Split:

Accuracy: 0.775

Precision: 0.744

Recall: 0.720

F1 score: 0.732

Confusion matrix:

```
[[176  40]
```

```
 [ 45 116]]
```

Cross Validation:

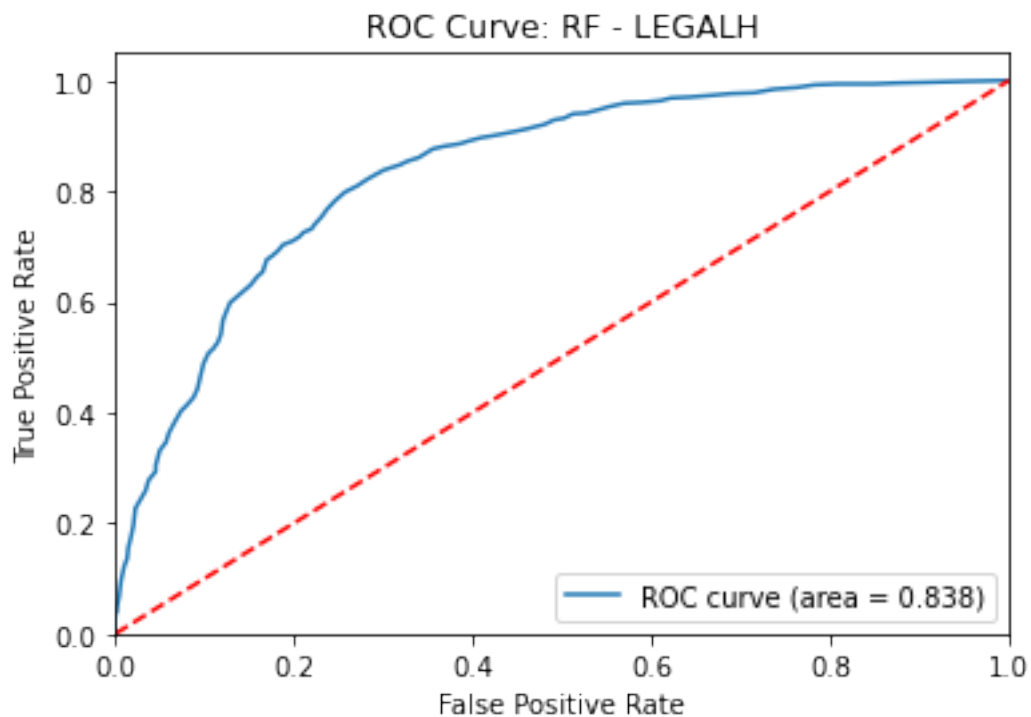
Mean Accuracy: 0.75597 +/- 0.05929

Mean Precision: 0.73003 +/- 0.05146

Mean Recall: 0.70476 +/- 0.01695

Mean F1 score: 0.69 +/- 0.11

Mean AUC Score: 0.83735 +/- 0.01755



--LSD

Train-Test Split:

Accuracy: 0.714

Precision: 0.658

Recall: 0.667

F1 score: 0.662

Confusion matrix:

```
[[163  55]
```

```
 [ 53 106]]
```

Cross Validation:

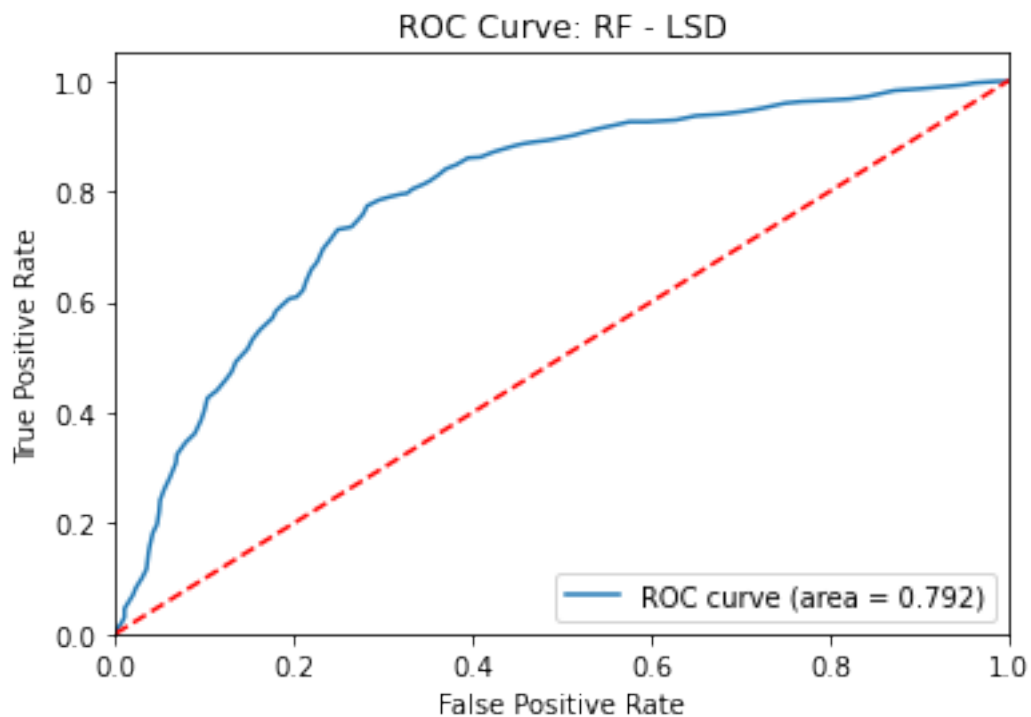
Mean Accuracy: 0.71512 +/- 0.03092

Mean Precision: 0.69872 +/- 0.02122

Mean Recall: 0.67280 +/- 0.02300

Mean F1 score: 0.67 +/- 0.09

Mean AUC Score: 0.79583 +/- 0.00940



--METH

Train-Test Split:

Accuracy: 0.764

Precision: 0.532

Recall: 0.355

F1 score: 0.426

Confusion matrix:

```
[[255  29]
```

```
 [ 60  33]]
```

Cross Validation:

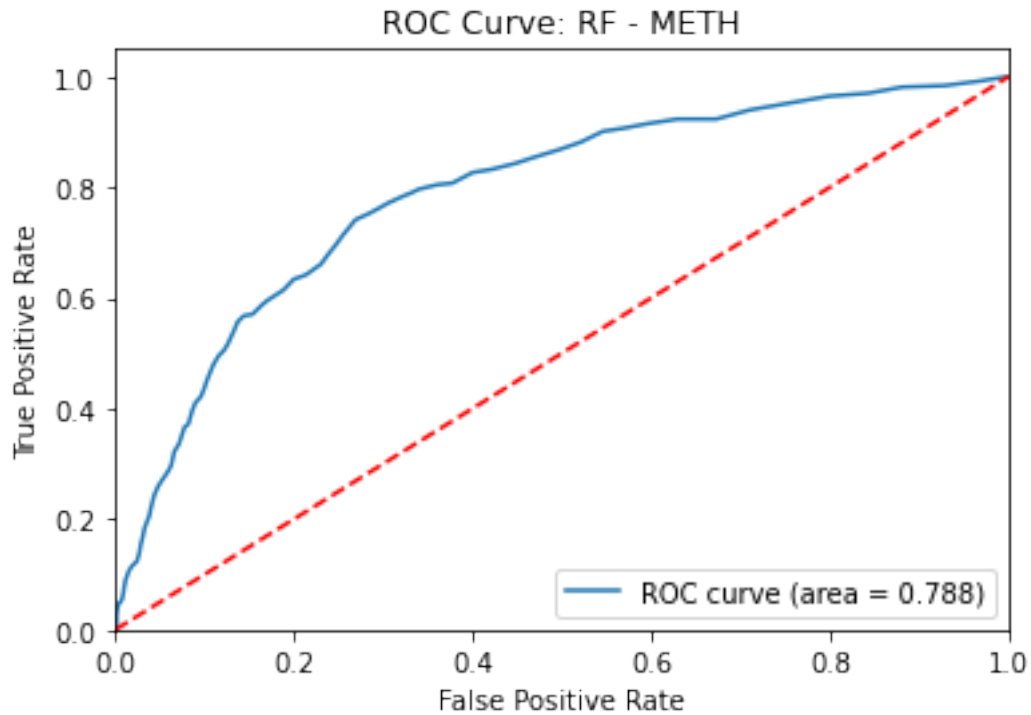
Mean Accuracy: 0.77347 +/- 0.02739

Mean Precision: 0.59167 +/- 0.04537

Mean Recall: 0.40502 +/- 0.04106

Mean F1 score: 0.42 +/- 0.12

Mean AUC Score: 0.78625 +/- 0.01973



--MUSHROOMS

Train-Test Split:

Accuracy: 0.700

Precision: 0.691

Recall: 0.710

F1 score: 0.700

Confusion matrix:

```
[[132  59]
```

```
 [ 54 132]]
```

Cross Validation:

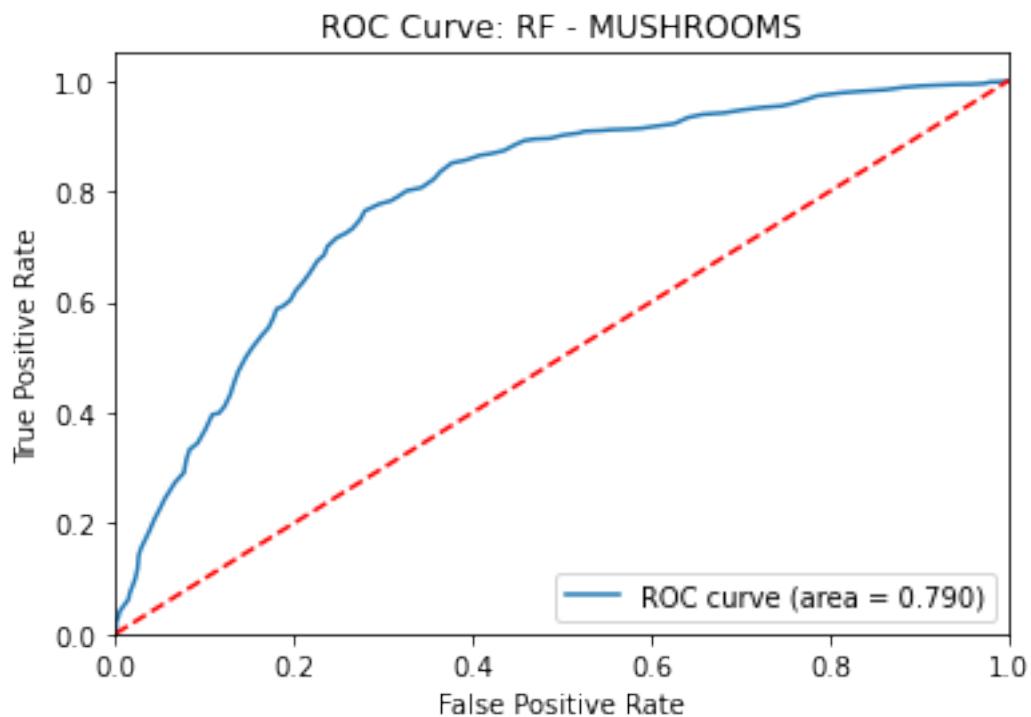
Mean Accuracy: 0.71989 +/- 0.05079

Mean Precision: 0.70836 +/- 0.02356

Mean Recall: 0.72525 +/- 0.04244

Mean F1 score: 0.70 +/- 0.09

Mean AUC Score: 0.79240 +/- 0.01895



--NICOTINE

Train-Test Split:

Accuracy: 0.753

Precision: 0.783

Recall: 0.934

F1 score: 0.852

Confusion matrix:

```
[[ 17  74]
```

```
 [ 19 267]]
```

Cross Validation:

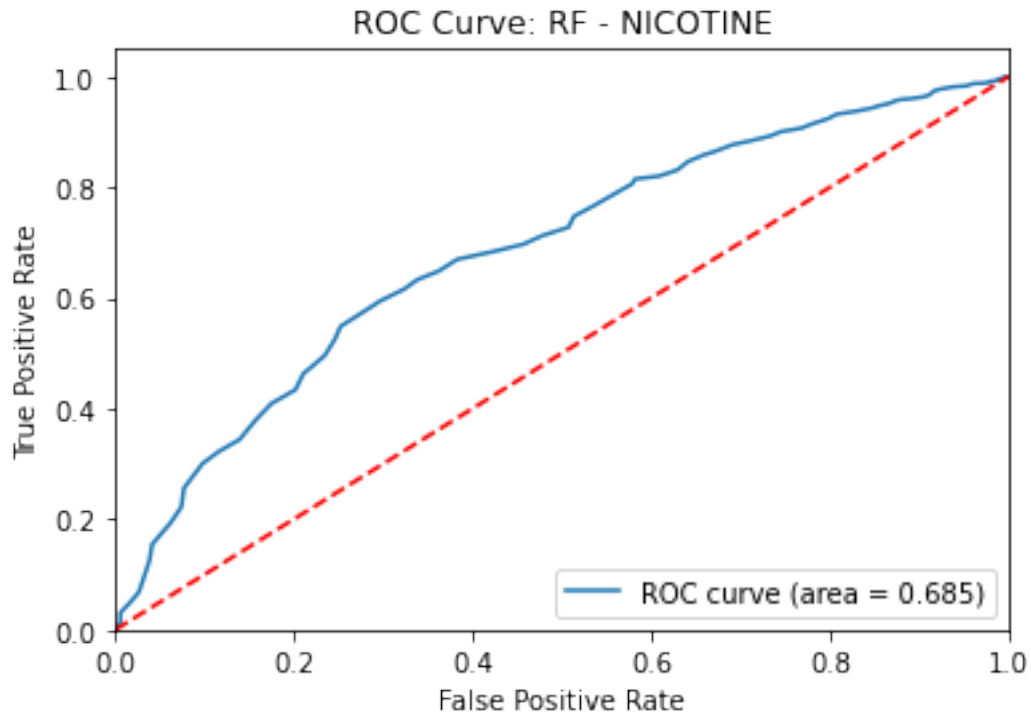
Mean Accuracy: 0.76233 +/- 0.02352

Mean Precision: 0.79460 +/- 0.00506

Mean Recall: 0.93764 +/- 0.01525

Mean F1 score: 0.86 +/- 0.02

Mean AUC Score: 0.67636 +/- 0.00931



## 12.6 Gradient Boosting Machine (GBM)

[17]:

```
df = pd.read_csv('data6.csv')
print("")

for col_index in range(9, len(df.columns)):

    columnName = df.columns[col_index]
    print()
    print("--" + (columnName))
    print()

    print('Train-Test Split:')

    X = df.iloc[:, 1:9]
    y = df[columnName]

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
↳ random_state=42)
```

```

# Create an instance of the model
model_1 = GradientBoostingClassifier()

# Train the model on the training data
model_1.fit(X_train, y_train)

# Make predictions on the test data
y_pred1 = model_1.predict(X_test)

# Calculate evaluation metrics
accuracy = accuracy_score(y_test, y_pred1)
precision = precision_score(y_test, y_pred1)
recall = recall_score(y_test, y_pred1)
f1 = f1_score(y_test, y_pred1)
conf_matrix = confusion_matrix(y_test, y_pred1)

# Print the evaluation metrics
print(f"Accuracy: {accuracy:.3f}")
print(f"Precision: {precision:.3f}")
print(f"Recall: {recall:.3f}")
print(f"F1 score: {f1:.3f}")
print("Confusion matrix:")
print(conf_matrix)
print()

N_FOLDS = 5
# Perform cross-validation predictions
y_scores = cross_val_predict(model_1, X_train, y_train, cv=N_FOLDS,
↪method="predict_proba")

fpr, tpr, thresholds = roc_curve(y_train, y_scores[:, 1])
roc_auc = roc_auc_score(y_train, y_scores[:, 1])

print()
print('Cross Validation:')
cv_accuracy = cross_val_score(model_1, X, y, cv=5, scoring='accuracy')
print('Mean Accuracy: %.5f +/- %.5f' % (np.mean(cv_accuracy), np.
↪std(cv_accuracy)))
precision_scores = cross_val_score(model_1, X_train, y_train, cv=N_FOLDS,
↪scoring="precision")
print('Mean Precision: %.5f +/- %.5f' % (np.mean(precision_scores), np.
↪std(precision_scores)))

```



```

    recall_scores = cross_val_score(model_1, X_train, y_train, cv=N_FOLDS,
↪scoring="recall")
    cv_f1 = cross_val_score(model_1, X, y, cv=5, scoring='f1')
    print('Mean Recall: %.5f +/- %.5f' % (np.mean(recall_scores), np.
↪std(recall_scores)))
    auc_scores = cross_val_score(model_1, X_train, y_train, cv=N_FOLDS,
↪scoring="roc_auc")
    print(f"Mean F1 score: {np.mean(cv_f1):.2f} +/- {np.std(cv_f1):.2f}")
    print('Mean AUC Score: %.5f +/- %.5f' % (np.mean(auc_scores), np.
↪std(auc_scores)))
    print()
    print()

    # Plot ROC curve
    plt.figure()
    plt.plot(fpr, tpr, label='ROC curve (area = %0.3f)' % roc_auc)
    plt.plot([0, 1], [0, 1], 'r--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    # Model
    plt.title('ROC Curve: GBM - ' + (columnName))
    plt.legend(loc="lower right")
    plt.show()

```

--AMPHET

Train-Test Split:

Accuracy: 0.637

Precision: 0.618

Recall: 0.635

F1 score: 0.627

Confusion matrix:

```
[[125  71]
```

```
 [ 66 115]]
```

Cross Validation:

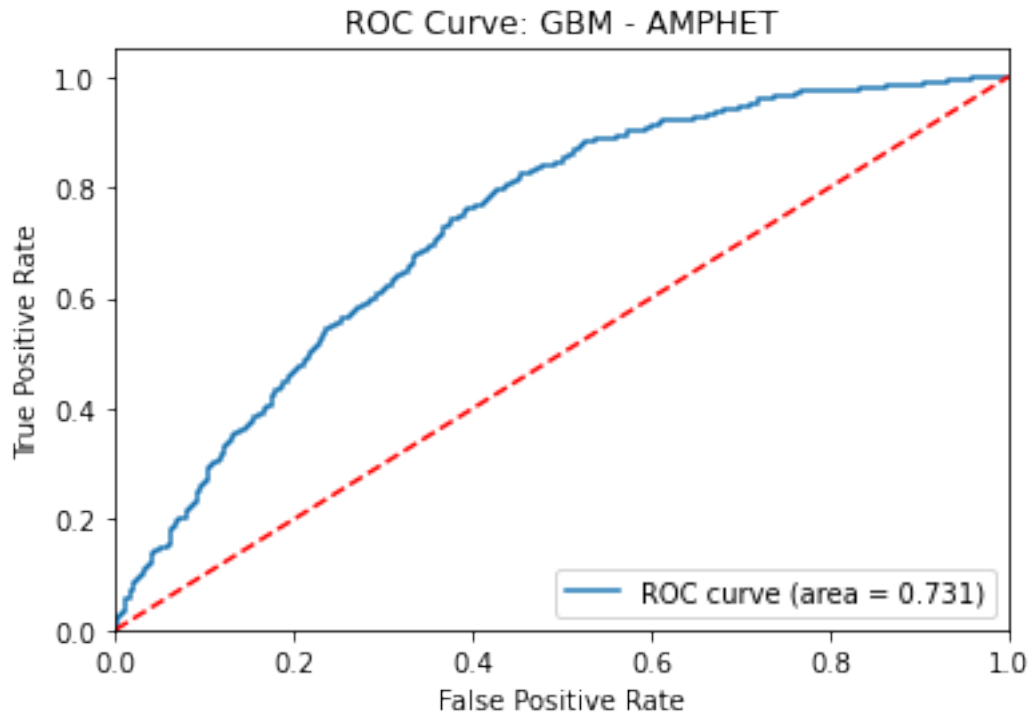
Mean Accuracy: 0.65995 +/- 0.03845

Mean Precision: 0.65082 +/- 0.01215

Mean Recall: 0.67994 +/- 0.04388

Mean F1 score: 0.65 +/- 0.08

Mean AUC Score: 0.73199 +/- 0.01400



--AMYL

Train-Test Split:

Accuracy: 0.721

Precision: 0.593

Recall: 0.302

F1 score: 0.400

Confusion matrix:

```
[[237  24]
```

```
 [ 81  35]]
```

Cross Validation:

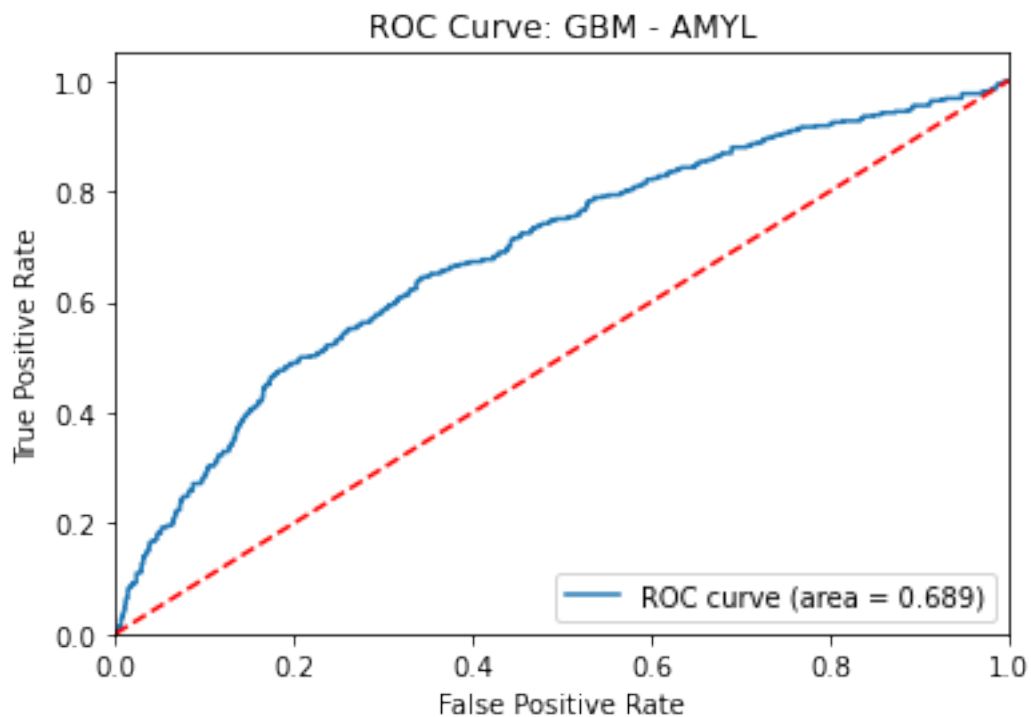
Mean Accuracy: 0.71300 +/- 0.01485

Mean Precision: 0.53879 +/- 0.06668

Mean Recall: 0.32737 +/- 0.05418

Mean F1 score: 0.43 +/- 0.07

Mean AUC Score: 0.69034 +/- 0.04116



--BENZOS

Train-Test Split:

Accuracy: 0.684

Precision: 0.663

Recall: 0.644

F1 score: 0.653

Confusion matrix:

```
[[146  57]
```

```
 [ 62 112]]
```

Cross Validation:

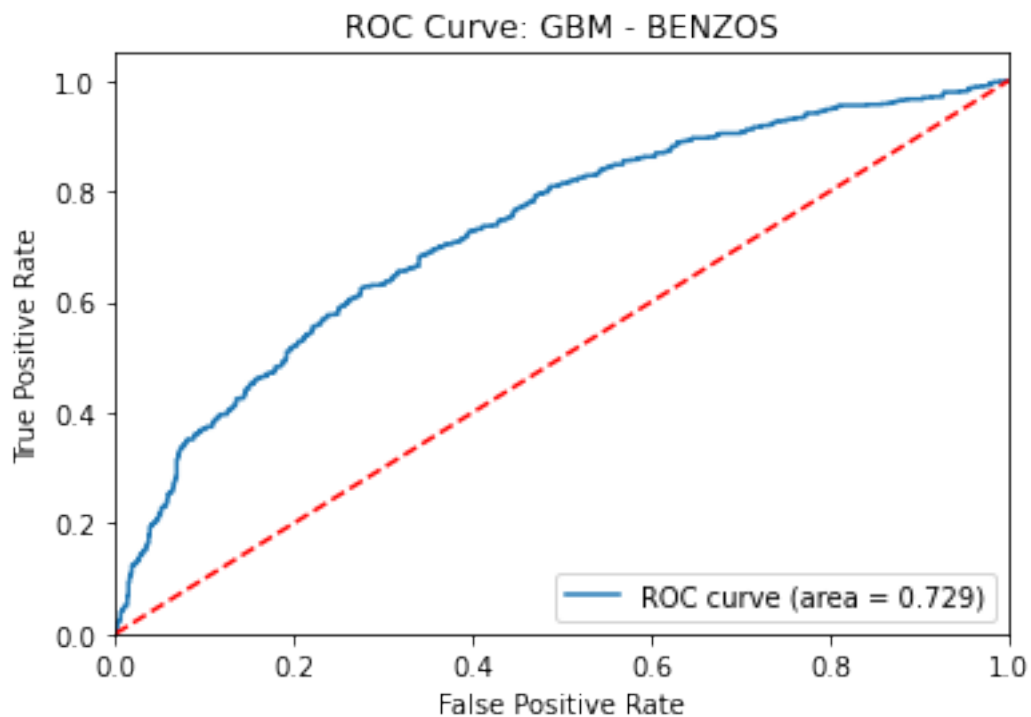
Mean Accuracy: 0.67003 +/- 0.02591

Mean Precision: 0.66814 +/- 0.02935

Mean Recall: 0.61313 +/- 0.04837

Mean F1 score: 0.63 +/- 0.08

Mean AUC Score: 0.73007 +/- 0.03234



--CANNABIS

Train-Test Split:

Accuracy: 0.785

Precision: 0.832

Recall: 0.904

F1 score: 0.867

Confusion matrix:

```
[[ 33  53]
```

```
 [ 28 263]]
```

Cross Validation:

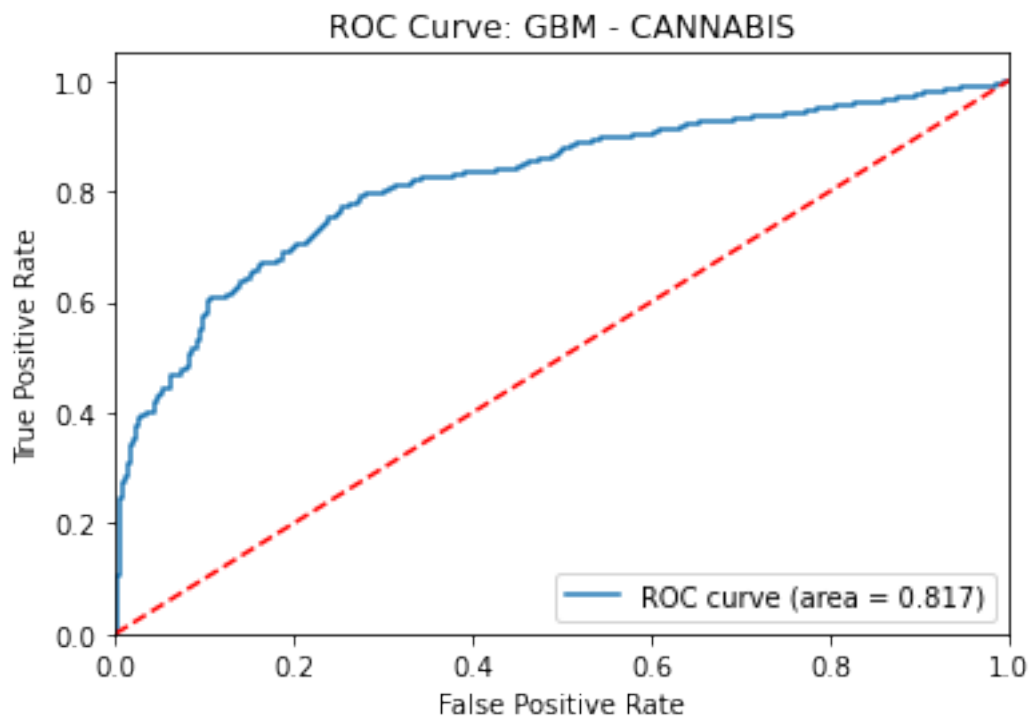
Mean Accuracy: 0.78727 +/- 0.04773

Mean Precision: 0.84580 +/- 0.01600

Mean Recall: 0.90173 +/- 0.03452

Mean F1 score: 0.87 +/- 0.04

Mean AUC Score: 0.81885 +/- 0.01931



--COKE

Train-Test Split:

Accuracy: 0.658

Precision: 0.582

Recall: 0.556

F1 score: 0.569

Confusion matrix:

```
[[163  61]
```

```
 [ 68  85]]
```

Cross Validation:

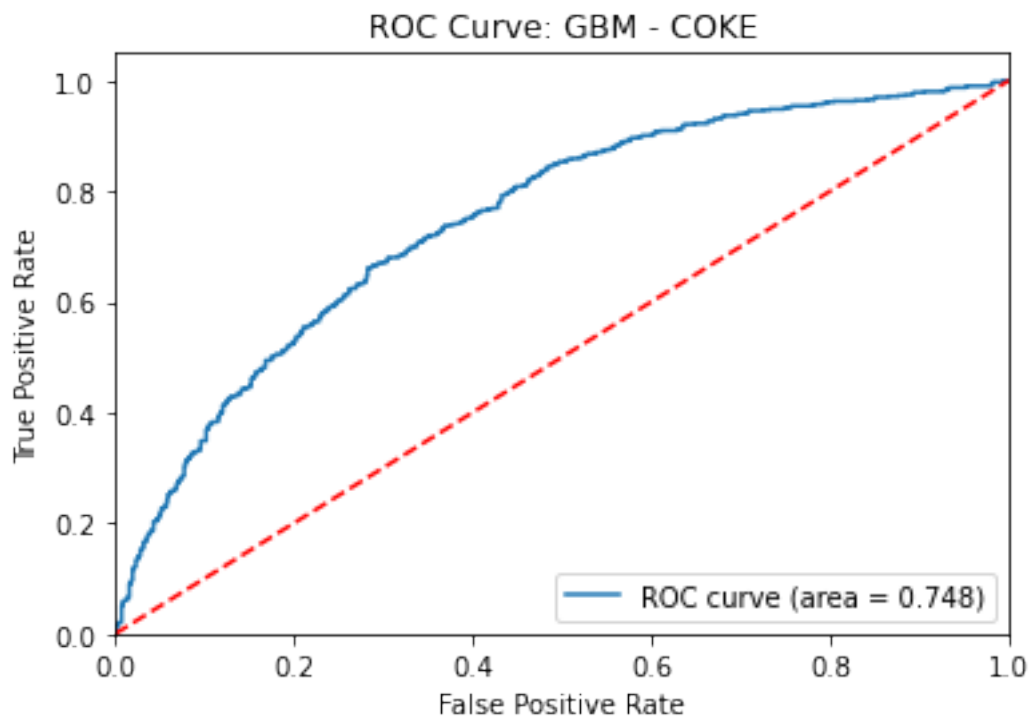
Mean Accuracy: 0.67851 +/- 0.04689

Mean Precision: 0.66156 +/- 0.01599

Mean Recall: 0.63115 +/- 0.03592

Mean F1 score: 0.63 +/- 0.07

Mean AUC Score: 0.74949 +/- 0.01524



--CRACK

Train-Test Split:

Accuracy: 0.875

Precision: 0.357

Recall: 0.116

F1 score: 0.175

Confusion matrix:

```
[[325  9]
```

```
 [ 38  5]]
```

Cross Validation:

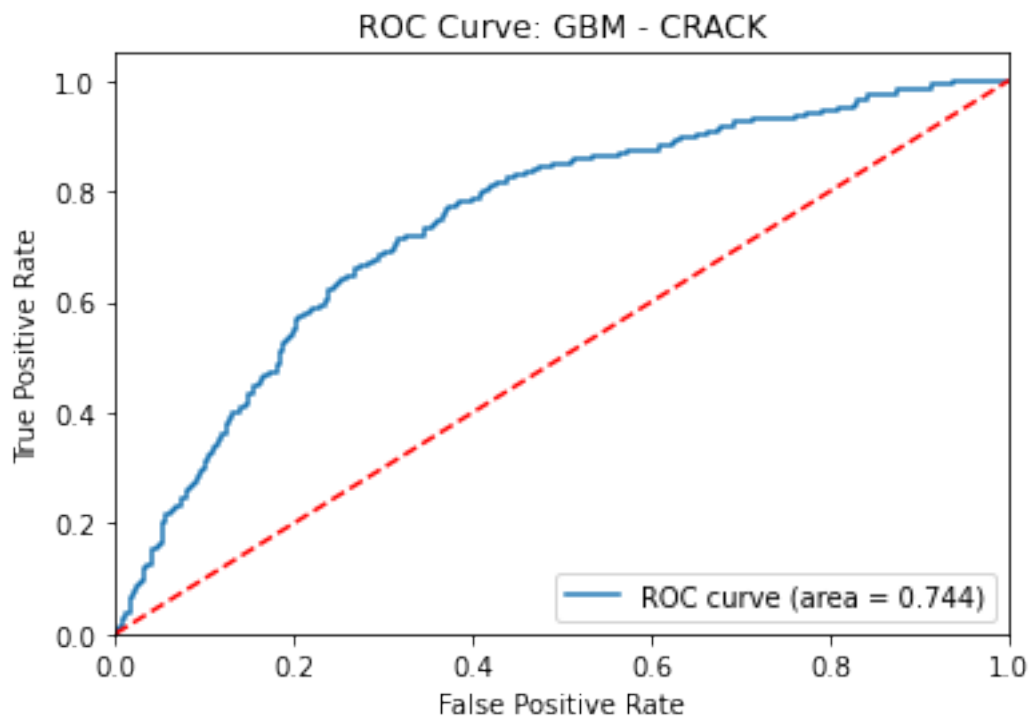
Mean Accuracy: 0.84668 +/- 0.01706

Mean Precision: 0.33647 +/- 0.08839

Mean Recall: 0.12558 +/- 0.04789

Mean F1 score: 0.14 +/- 0.06

Mean AUC Score: 0.74495 +/- 0.03273



--ECSTASY

Train-Test Split:

Accuracy: 0.698

Precision: 0.652

Recall: 0.698

F1 score: 0.674

Confusion matrix:

```
[[145  63]
```

```
 [ 51 118]]
```

Cross Validation:

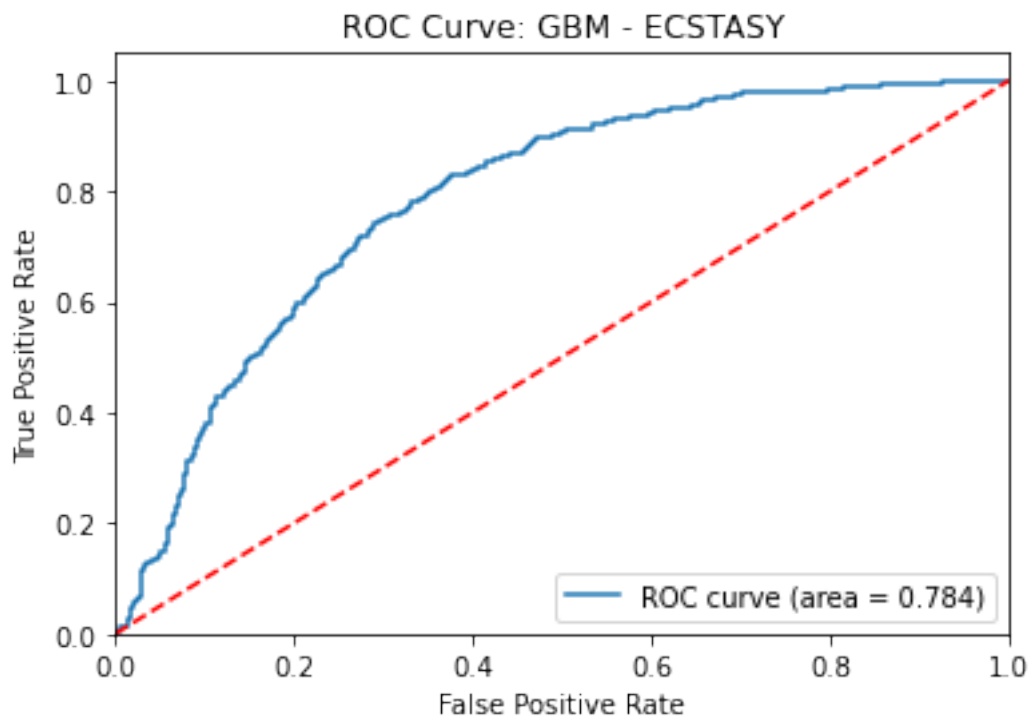
Mean Accuracy: 0.70027 +/- 0.05541

Mean Precision: 0.69288 +/- 0.01954

Mean Recall: 0.69640 +/- 0.04845

Mean F1 score: 0.67 +/- 0.10

Mean AUC Score: 0.78548 +/- 0.00790



--HEROIN

Train-Test Split:

Accuracy: 0.849

Precision: 0.389

Recall: 0.132

F1 score: 0.197

Confusion matrix:

```
[[313  11]
```

```
 [ 46   7]]
```

Cross Validation:

Mean Accuracy: 0.83820 +/- 0.02294

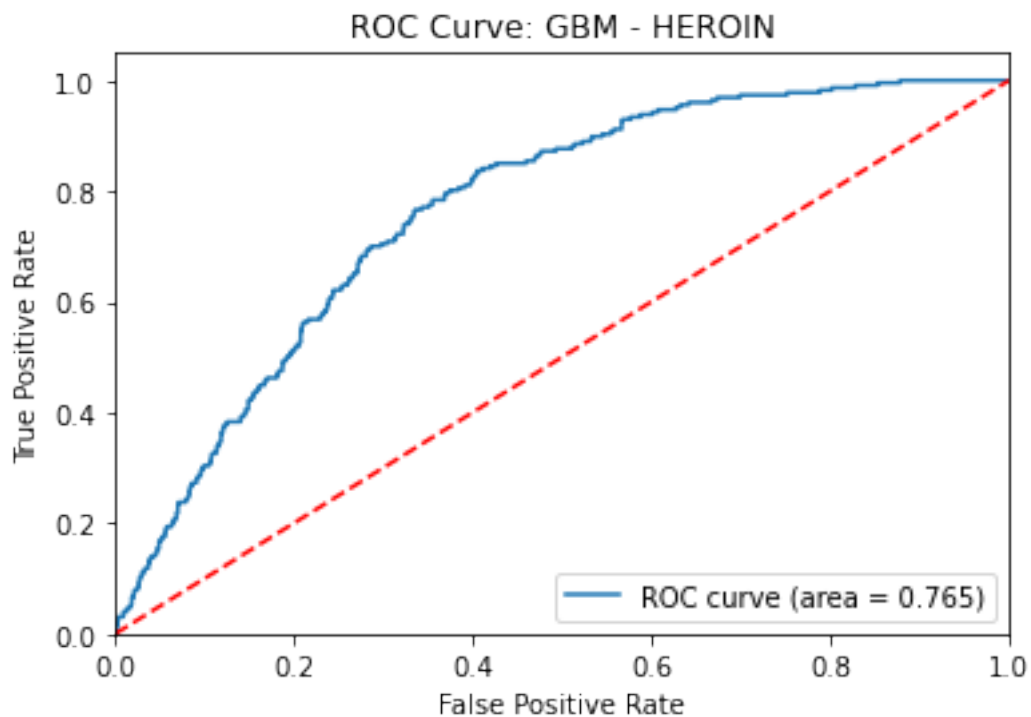
Mean Precision: 0.36542 +/- 0.13737

Mean Recall: 0.14155 +/- 0.07830

Mean F1 score: 0.21 +/- 0.06

Mean AUC Score: 0.76746 +/- 0.04192





--KETAMINE

Train-Test Split:

Accuracy: 0.775

Precision: 0.167

Recall: 0.058

F1 score: 0.086

Confusion matrix:

```
[[288  20]
```

```
 [ 65   4]]
```

Cross Validation:

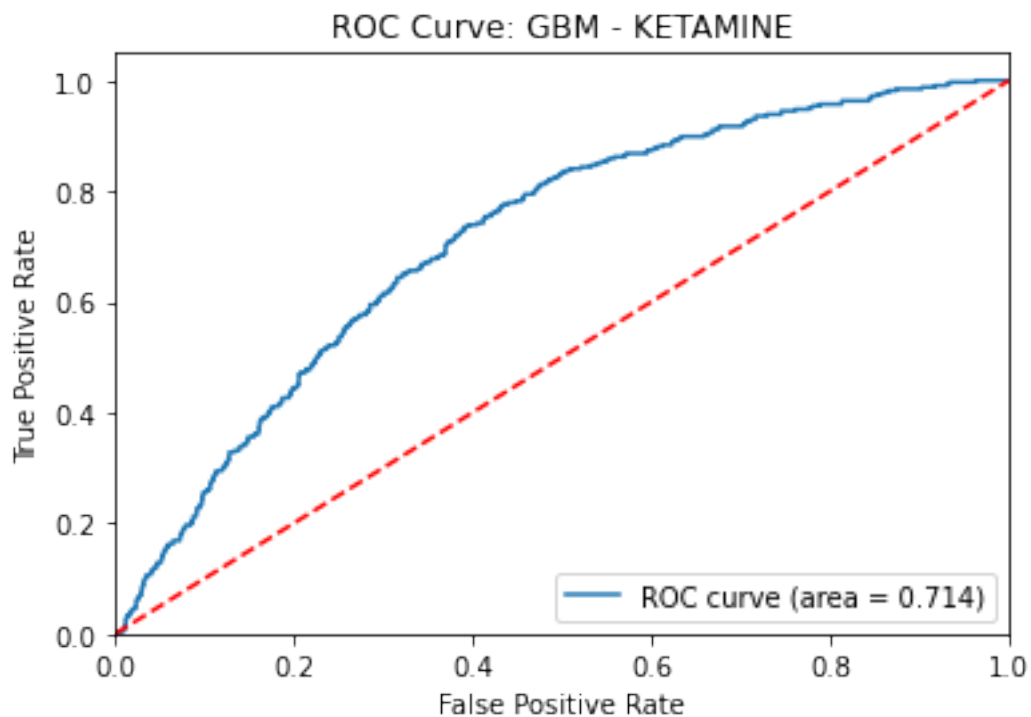
Mean Accuracy: 0.76817 +/- 0.01632

Mean Precision: 0.43120 +/- 0.05176

Mean Recall: 0.11967 +/- 0.03439

Mean F1 score: 0.12 +/- 0.04

Mean AUC Score: 0.71354 +/- 0.02057



--LEGALH

Train-Test Split:

Accuracy: 0.772

Precision: 0.727

Recall: 0.745

F1 score: 0.736

Confusion matrix:

```
[[171  45]
```

```
 [ 41 120]]
```

Cross Validation:

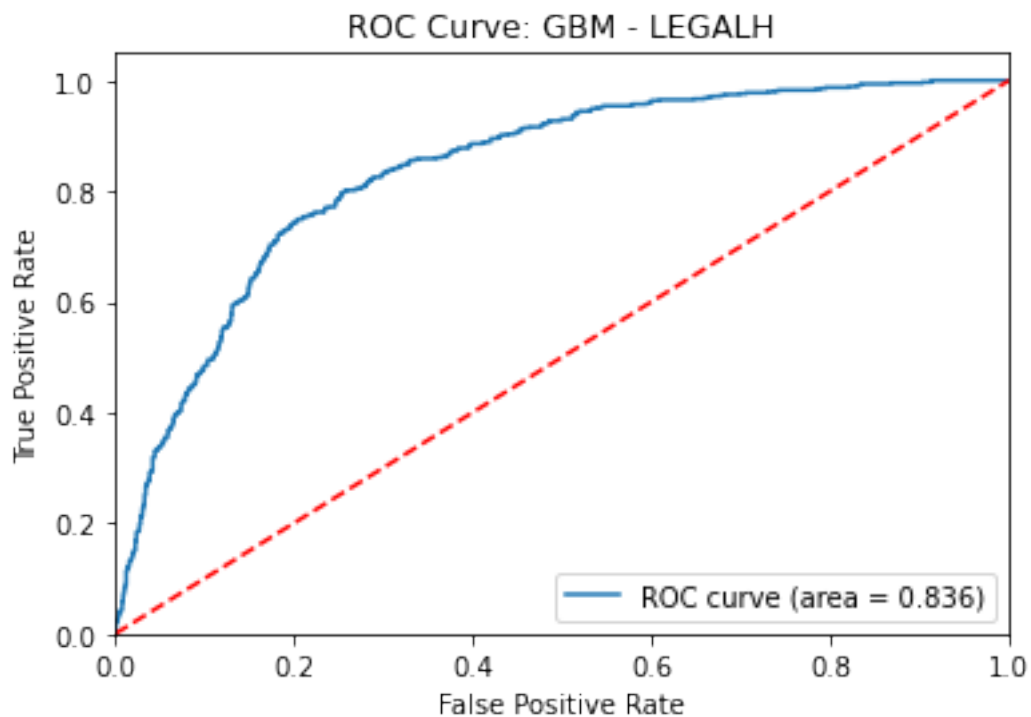
Mean Accuracy: 0.75225 +/- 0.06544

Mean Precision: 0.73863 +/- 0.04611

Mean Recall: 0.71905 +/- 0.04157

Mean F1 score: 0.69 +/- 0.12

Mean AUC Score: 0.83598 +/- 0.01473



--LSD

Train-Test Split:

Accuracy: 0.708

Precision: 0.648

Recall: 0.673

F1 score: 0.660

Confusion matrix:

```
[[160  58]
```

```
 [ 52 107]]
```

Cross Validation:

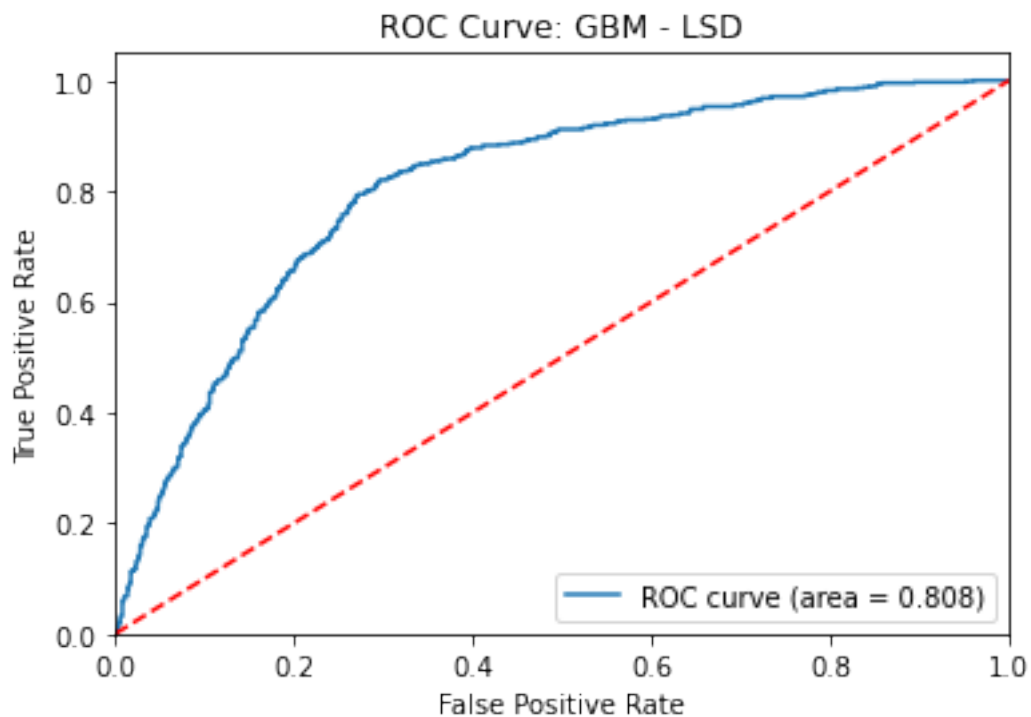
Mean Accuracy: 0.73528 +/- 0.04324

Mean Precision: 0.70062 +/- 0.02280

Mean Recall: 0.70933 +/- 0.02265

Mean F1 score: 0.68 +/- 0.10

Mean AUC Score: 0.80937 +/- 0.01837



--METH

Train-Test Split:

Accuracy: 0.759

Precision: 0.516

Recall: 0.355

F1 score: 0.420

Confusion matrix:

```
[[253  31]
```

```
 [ 60  33]]
```

Cross Validation:

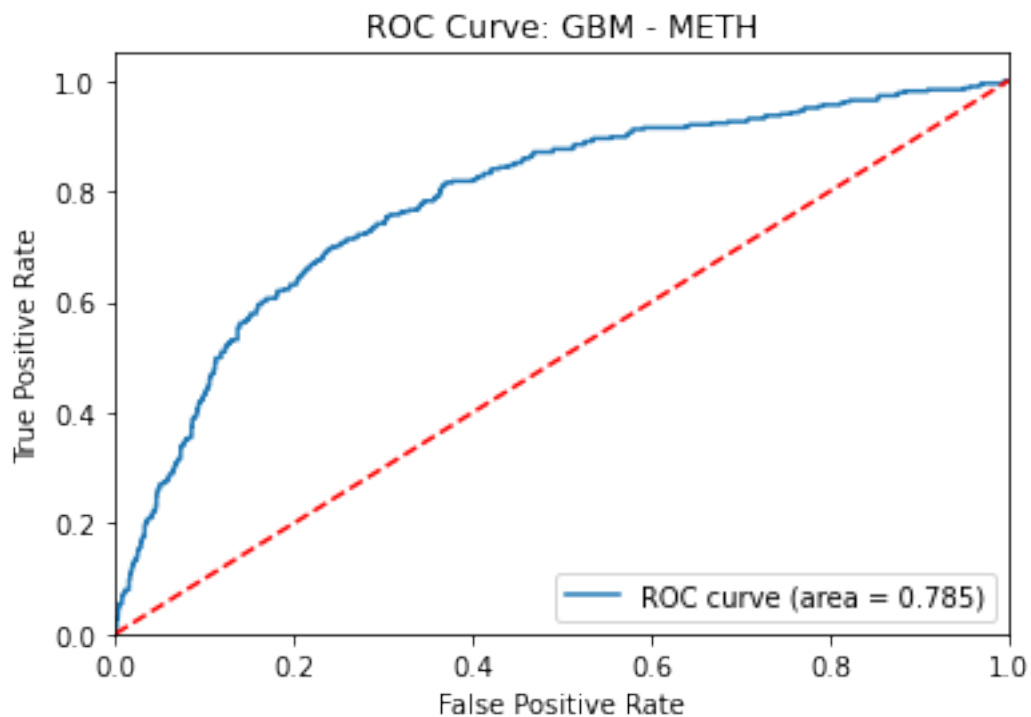
Mean Accuracy: 0.76976 +/- 0.03095

Mean Precision: 0.57880 +/- 0.03699

Mean Recall: 0.44627 +/- 0.04138

Mean F1 score: 0.43 +/- 0.13

Mean AUC Score: 0.78632 +/- 0.01596



--MUSHROOMS

Train-Test Split:

Accuracy: 0.708

Precision: 0.704

Recall: 0.704

F1 score: 0.704

Confusion matrix:

```
[[136  55]
```

```
 [ 55 131]]
```

Cross Validation:

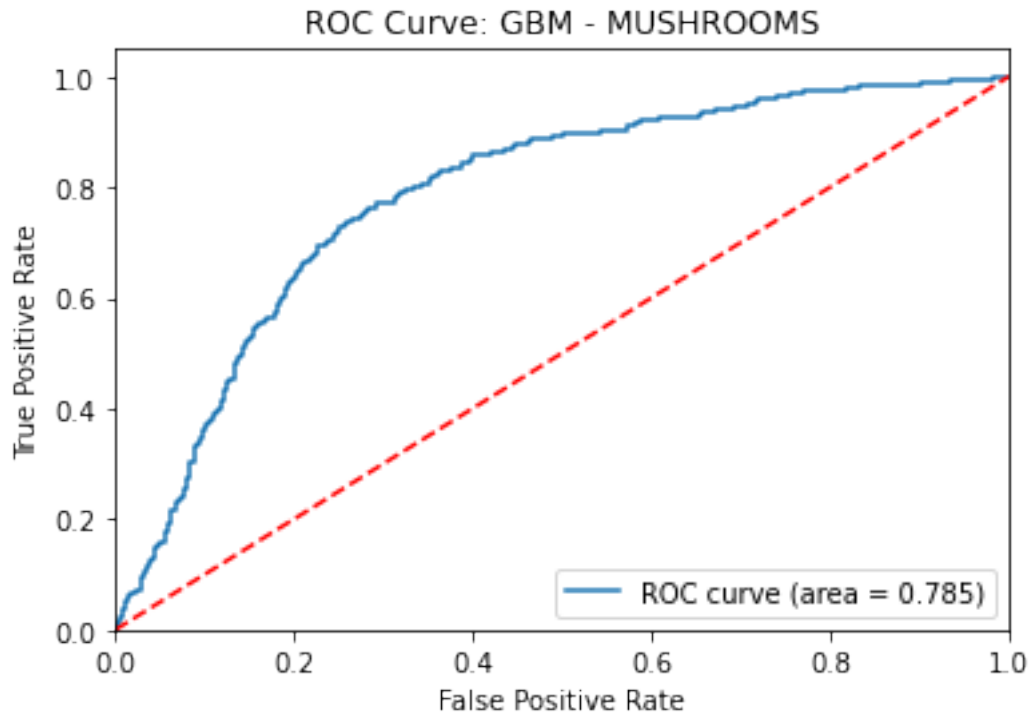
Mean Accuracy: 0.71034 +/- 0.05773

Mean Precision: 0.72243 +/- 0.02436

Mean Recall: 0.73363 +/- 0.03389

Mean F1 score: 0.69 +/- 0.10

Mean AUC Score: 0.78571 +/- 0.01360



--NICOTINE

Train-Test Split:

Accuracy: 0.772

Precision: 0.798

Recall: 0.937

F1 score: 0.862

Confusion matrix:

```
[[ 23  68]
```

```
 [ 18 268]]
```

Cross Validation:

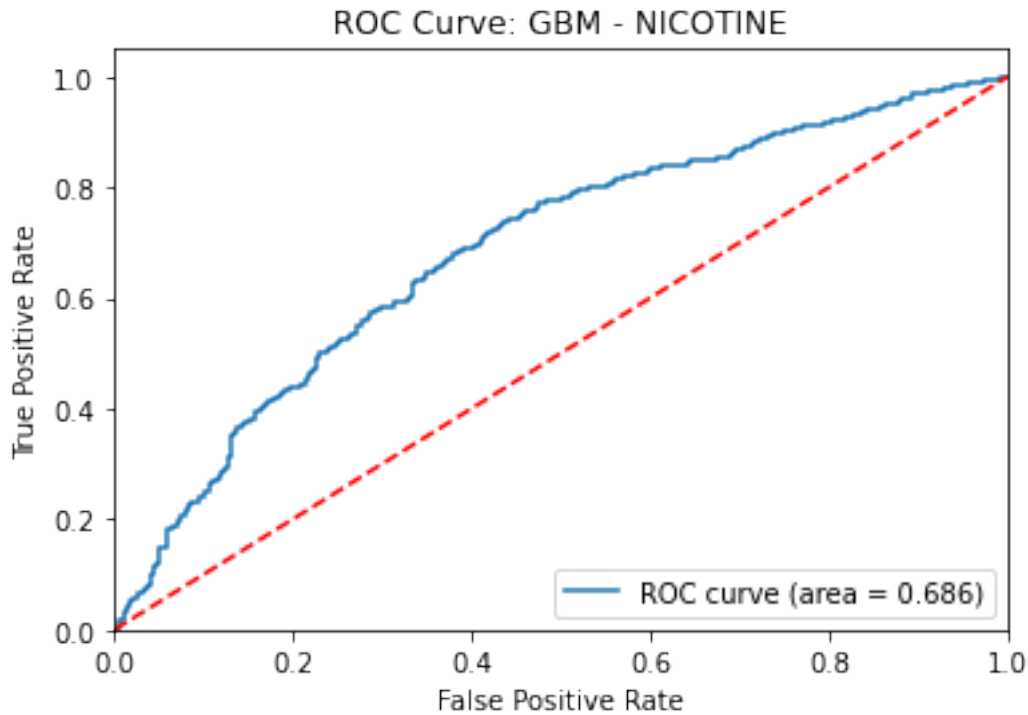
Mean Accuracy: 0.76021 +/- 0.02860

Mean Precision: 0.79453 +/- 0.00338

Mean Recall: 0.93764 +/- 0.02995

Mean F1 score: 0.86 +/- 0.02

Mean AUC Score: 0.68650 +/- 0.02916



## 12.7 SVM

```
[19]: from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_val_predict
from sklearn.metrics import make_scorer
df = pd.read_csv('data6.csv')

print("")

for col_index in range(9, len(df.columns)):

    columnName = df.columns[col_index]
    print()
    print("--" + (columnName))
```

```

X = df.iloc[:, 1:9]
y = df[columnName]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
↪random_state=42)

# Create an instance of the model
model_1 = SVC(probability=True)

# Train the model on the training data
model_1.fit(X_train, y_train)

# Make predictions on the test data
y_pred1 = model_1.predict(X_test)

# Calculate evaluation metrics
accuracy = accuracy_score(y_test, y_pred1)
precision = precision_score(y_test, y_pred1, zero_division=1)
recall = recall_score(y_test, y_pred1)
f1 = f1_score(y_test, y_pred1)
conf_matrix = confusion_matrix(y_test, y_pred1)

# Print the evaluation metrics
print()
print('Train-Test Split')
print(f"Accuracy: {accuracy:.3f}")
print(f"Precision: {precision:.3f}")
print(f"Recall: {recall:.3f}")
print(f"F1 score: {f1:.3f}")
print("Confusion matrix:")
print(conf_matrix)

# Perform cross-validation predictions
y_scores = cross_val_predict(model_1, X_train, y_train, cv=5,
↪method="predict_proba")

# Binary classification (y_scores[:, 1] for positive class)
fpr, tpr, thresholds = roc_curve(y_train, y_scores[:, 1])
roc_auc = roc_auc_score(y_train, y_scores[:, 1])

def custom_precision(y_true, y_pred):

```



```

    try:
        return precision_score(y_true, y_pred)
    except ZeroDivisionError:
        return 0.0

print()
print('Cross Validation:')
cv_accuracy = cross_val_score(model_1, X, y, cv=5, scoring='accuracy')
print('Mean Accuracy: %.5f +/- %.5f' % (np.mean(cv_accuracy), np.
↪std(cv_accuracy)))

# Create a scorer using the custom precision scorer
precision_scorer = make_scorer(custom_precision)

precision_scores = cross_val_score(model_1, X_train, y_train, cv=N_FOLDS,
↪scoring=make_scorer(precision_score, zero_division=1))
#precision_scores = cross_val_score(model_1, X_train, y_train, cv=N_FOLDS,
↪scoring="precision")
print('Mean Precision: %.5f +/- %.5f' % (np.mean(precision_scores), np.
↪std(precision_scores)))
recall_scores = cross_val_score(model_1, X_train, y_train, cv=N_FOLDS,
↪scoring="recall")
cv_f1 = cross_val_score(model_1, X, y, cv=5, scoring='f1')

print('Mean Recall: %.5f +/- %.5f' % (np.mean(recall_scores), np.
↪std(recall_scores)))
print(f"Mean F1 score: {np.mean(cv_f1):.2f} +/- {np.std(cv_f1):.2f}")
auc_scores = cross_val_score(model_1, X_train, y_train, cv=N_FOLDS,
↪scoring="roc_auc")
print('Mean AUC Score: %.5f +/- %.5f' % (np.mean(auc_scores), np.
↪std(auc_scores)))
print()
print()

# Plot ROC curve
plt.figure()
plt.plot(fpr, tpr, label='ROC curve (area = %0.3f)' % roc_auc)
plt.plot([0, 1], [0, 1], 'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve: SVM - ' + (columnName))
plt.legend(loc="lower right")
plt.show()

```

--AMPHET

Train-Test Split

Accuracy: 0.660

Precision: 0.640

Recall: 0.669

F1 score: 0.654

Confusion matrix:

[[128 68]

[ 60 121]]

Cross Validation:

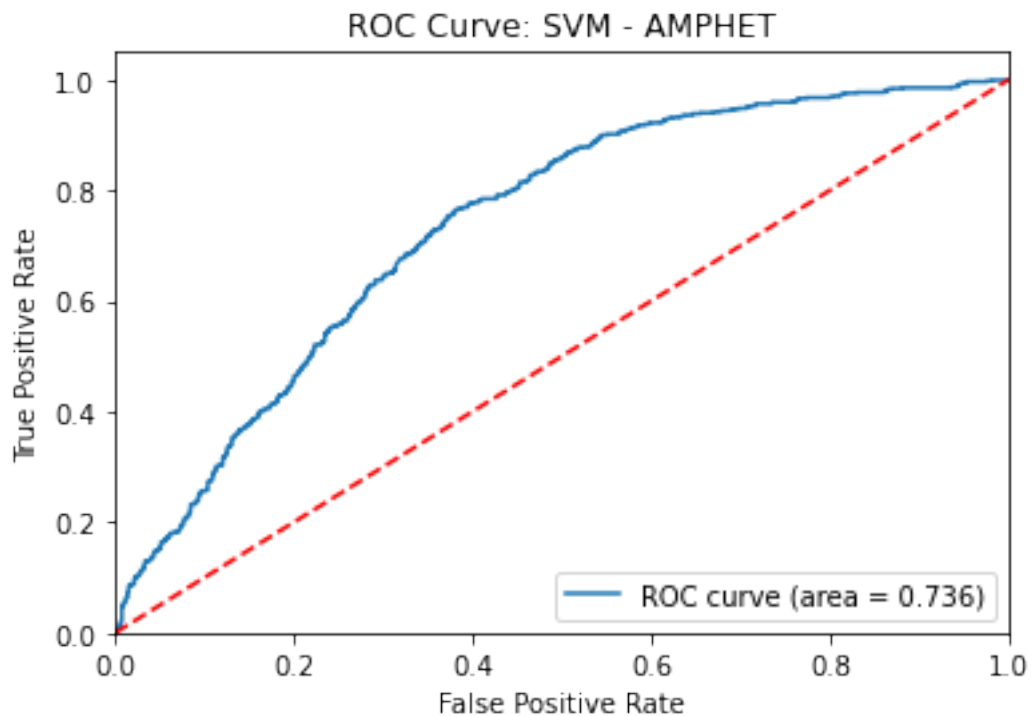
Mean Accuracy: 0.65040 +/- 0.04162

Mean Precision: 0.65853 +/- 0.01299

Mean Recall: 0.70333 +/- 0.02572

Mean F1 score: 0.64 +/- 0.10

Mean AUC Score: 0.73752 +/- 0.01005



--AMYL

Train-Test Split

Accuracy: 0.711

Precision: 0.569

Recall: 0.250

F1 score: 0.347

Confusion matrix:

```
[[239  22]
```

```
 [ 87  29]]
```

Cross Validation:

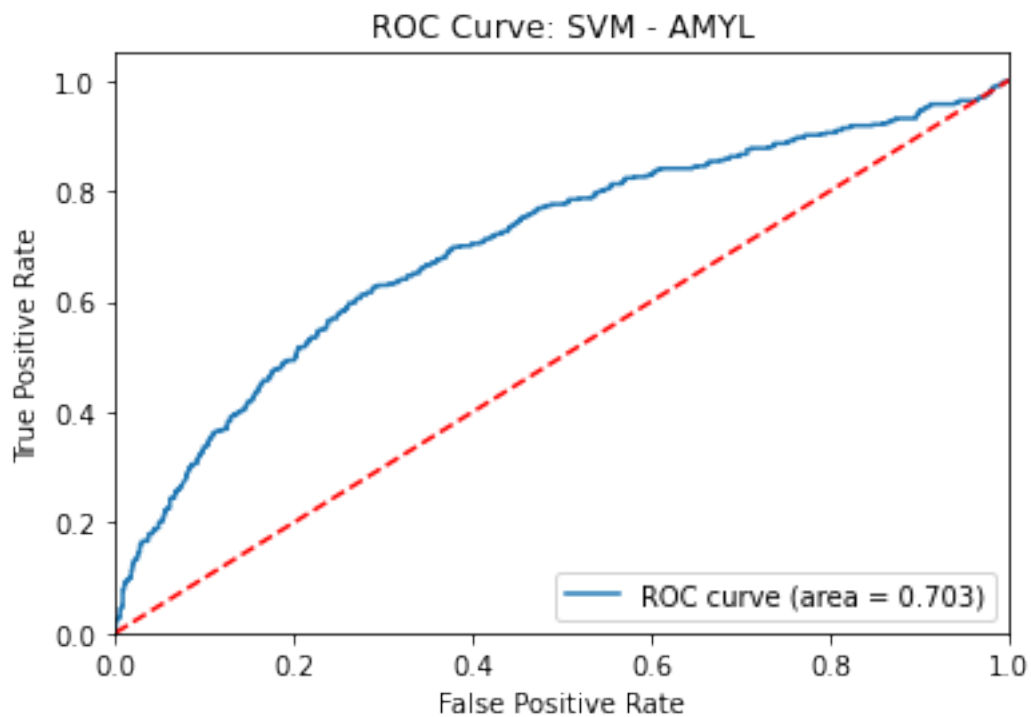
Mean Accuracy: 0.71724 +/- 0.00943

Mean Precision: 0.62086 +/- 0.06589

Mean Recall: 0.27803 +/- 0.03152

Mean F1 score: 0.37 +/- 0.07

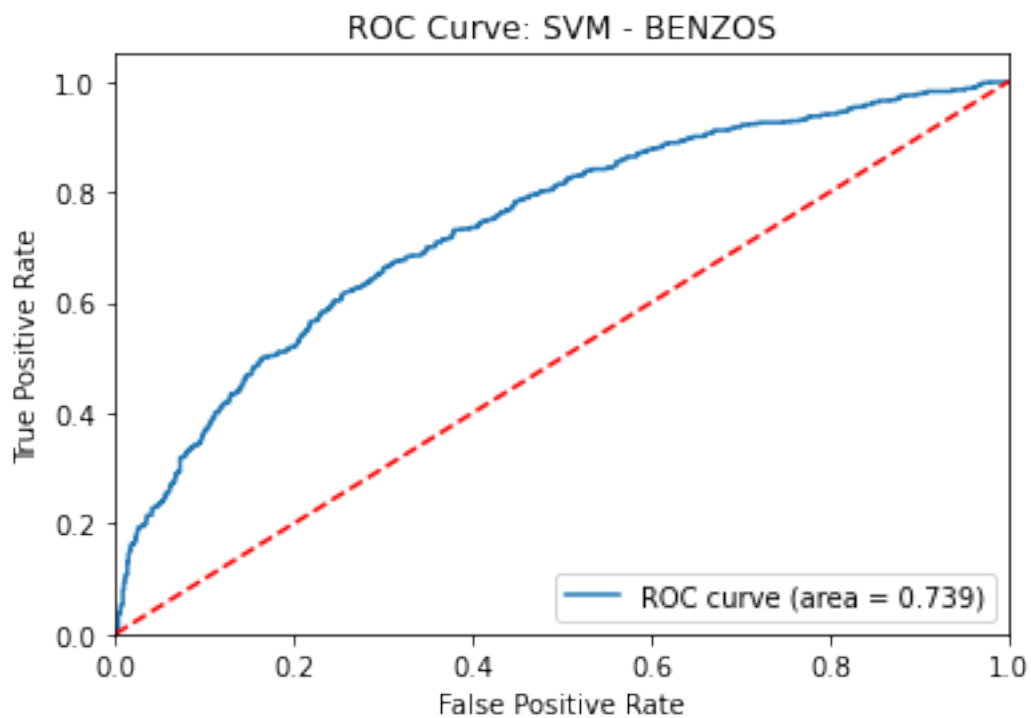
Mean AUC Score: 0.70746 +/- 0.04228



--BENZOS

Train-Test Split  
Accuracy: 0.690  
Precision: 0.675  
Recall: 0.632  
F1 score: 0.653  
Confusion matrix:  
[[150 53]  
 [ 64 110]]

Cross Validation:  
Mean Accuracy: 0.67215 +/- 0.03387  
Mean Precision: 0.67881 +/- 0.03479  
Mean Recall: 0.62019 +/- 0.04500  
Mean F1 score: 0.62 +/- 0.11  
Mean AUC Score: 0.74187 +/- 0.03282

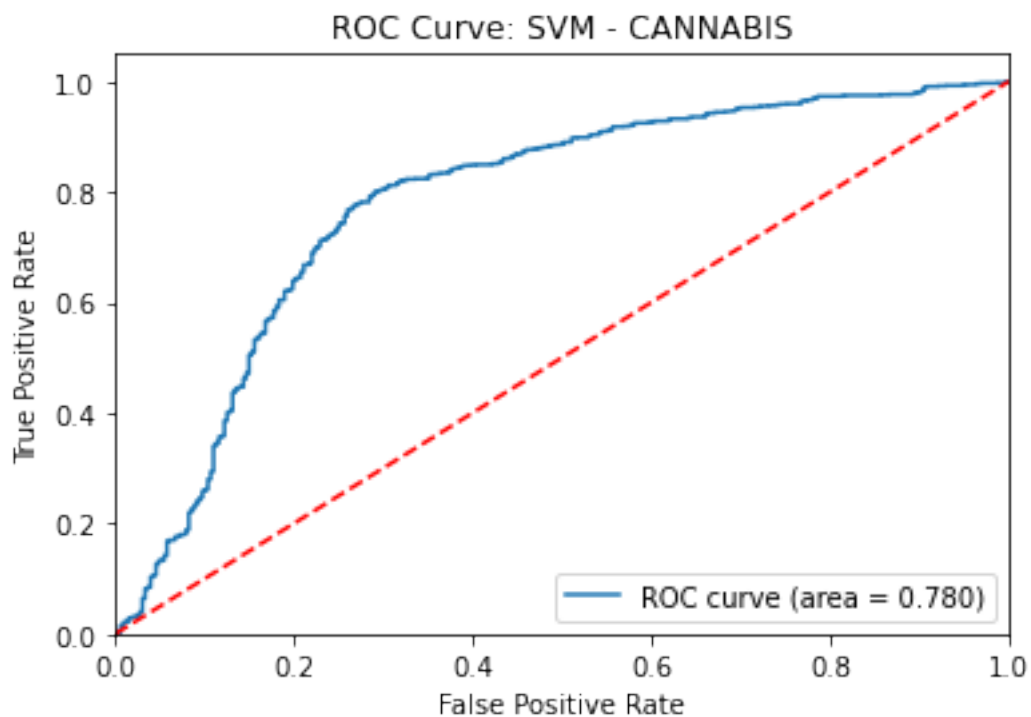


--CANNABIS

Train-Test Split  
Accuracy: 0.817

Precision: 0.836  
Recall: 0.948  
F1 score: 0.889  
Confusion matrix:  
[[ 32 54]  
 [ 15 276]]

Cross Validation:  
Mean Accuracy: 0.80424 +/- 0.03255  
Mean Precision: 0.83924 +/- 0.01111  
Mean Recall: 0.93900 +/- 0.02439  
Mean F1 score: 0.88 +/- 0.03  
Mean AUC Score: 0.78114 +/- 0.01495



--COKE

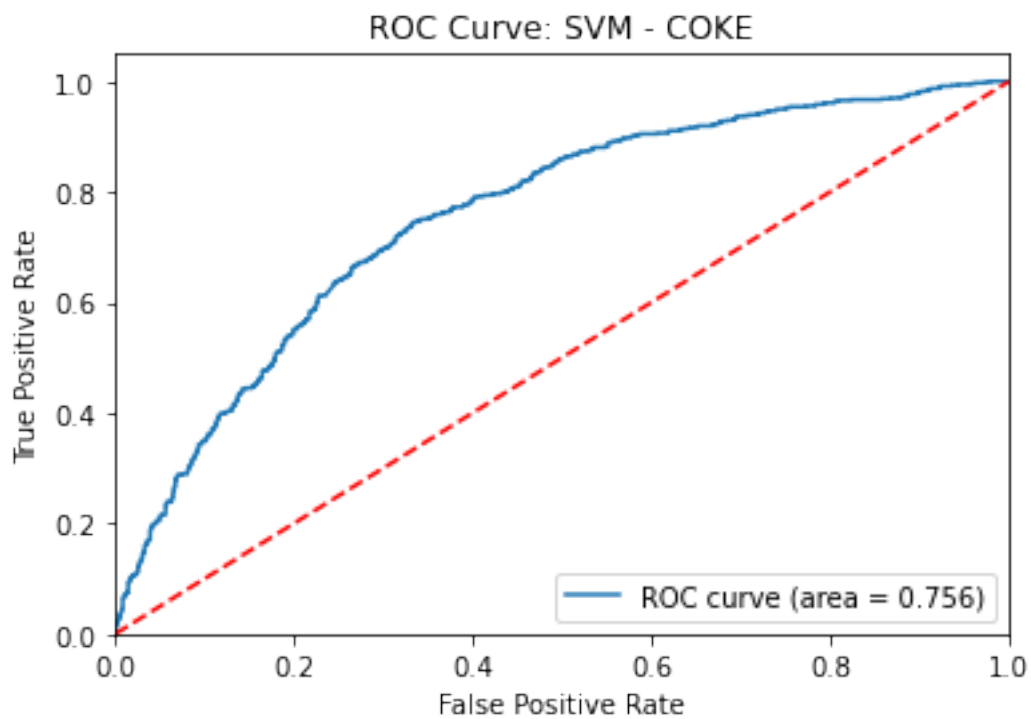
Train-Test Split  
Accuracy: 0.650  
Precision: 0.570  
Recall: 0.562  
F1 score: 0.566

Confusion matrix:

```
[[159  65]
 [ 67  86]]
```

Cross Validation:

Mean Accuracy: 0.66419 +/- 0.04115  
Mean Precision: 0.68461 +/- 0.03270  
Mean Recall: 0.63833 +/- 0.03230  
Mean F1 score: 0.60 +/- 0.10  
Mean AUC Score: 0.75762 +/- 0.01900



--CRACK

Train-Test Split

Accuracy: 0.886

Precision: 1.000

Recall: 0.000

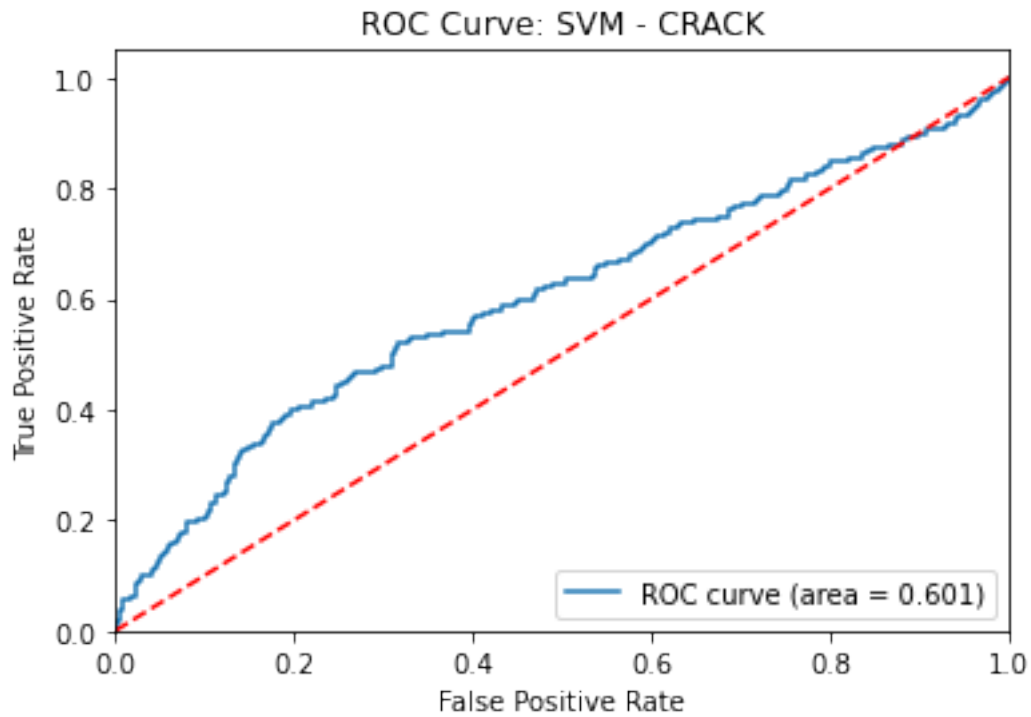
F1 score: 0.000

Confusion matrix:

```
[[334  0]
 [ 43  0]]
```

Cross Validation:

Mean Accuracy: 0.86313 +/- 0.00130  
Mean Precision: 1.00000 +/- 0.00000  
Mean Recall: 0.00000 +/- 0.00000  
Mean F1 score: 0.00 +/- 0.00  
Mean AUC Score: 0.59975 +/- 0.04489



--ECSTASY

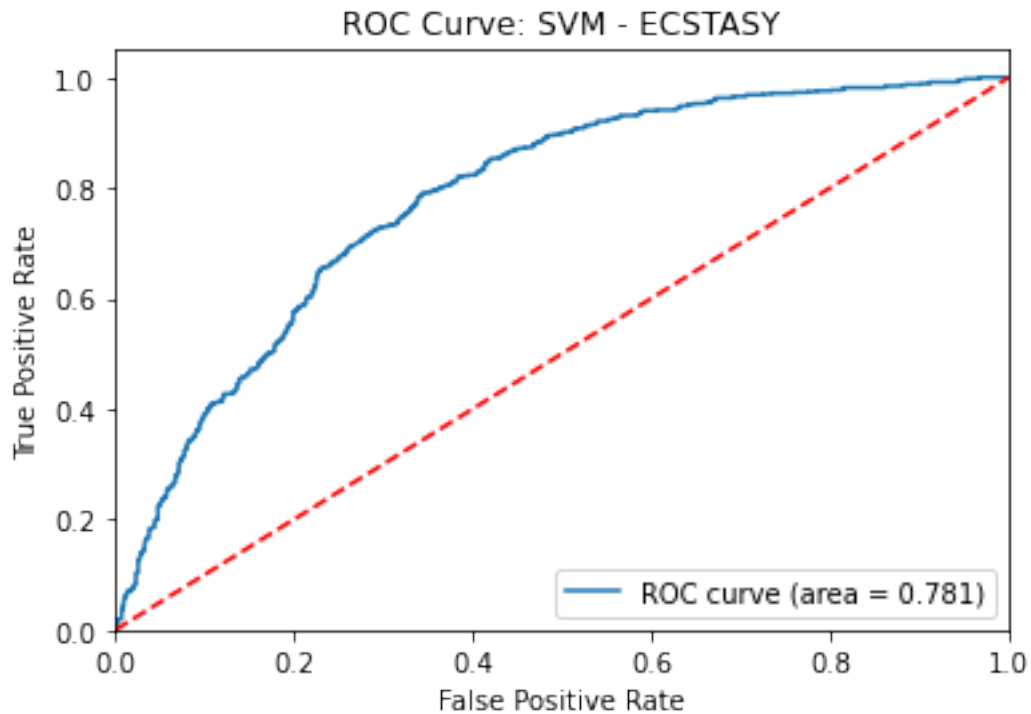
Train-Test Split

Accuracy: 0.716  
Precision: 0.663  
Recall: 0.746  
F1 score: 0.702  
Confusion matrix:  
[[144 64]  
 [ 43 126]]

Cross Validation:

Mean Accuracy: 0.70398 +/- 0.05825

Mean Precision: 0.68376 +/- 0.01440  
Mean Recall: 0.72230 +/- 0.02790  
Mean F1 score: 0.68 +/- 0.10  
Mean AUC Score: 0.78207 +/- 0.01328



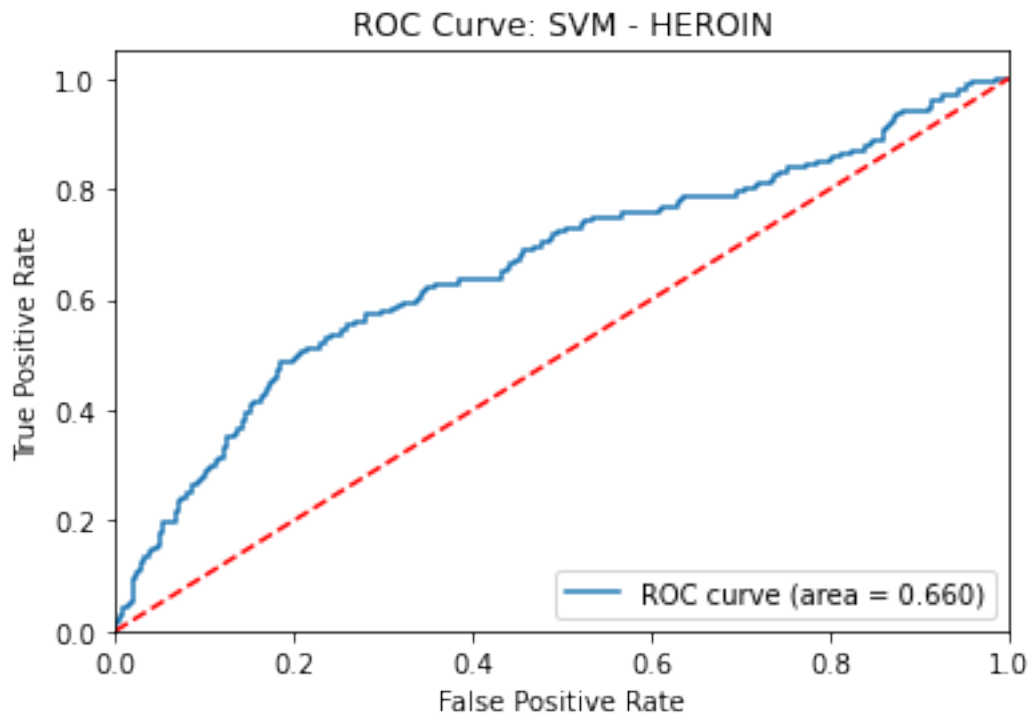
--HEROIN

Train-Test Split  
Accuracy: 0.859  
Precision: 1.000  
Recall: 0.000  
F1 score: 0.000  
Confusion matrix:  
[[324 0]  
 [ 53 0]]

Cross Validation:  
Mean Accuracy: 0.85040 +/- 0.00212  
Mean Precision: 0.56667 +/- 0.38873  
Mean Recall: 0.00879 +/- 0.01077  
Mean F1 score: 0.03 +/- 0.03



Mean AUC Score: 0.66846 +/- 0.06770



--KETAMINE

Train-Test Split

Accuracy: 0.817

Precision: 1.000

Recall: 0.000

F1 score: 0.000

Confusion matrix:

```
[[308  0]
```

```
 [ 69  0]]
```

Cross Validation:

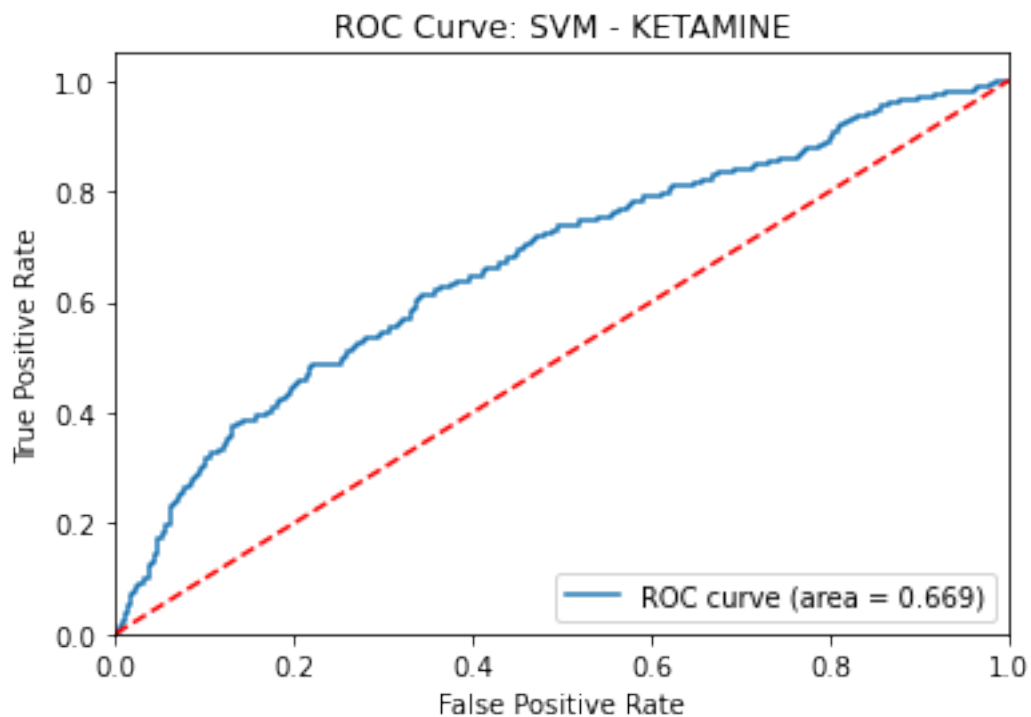
Mean Accuracy: 0.78780 +/- 0.00556

Mean Precision: 0.50000 +/- 0.44721

Mean Recall: 0.00308 +/- 0.00615

Mean F1 score: 0.03 +/- 0.03

Mean AUC Score: 0.66844 +/- 0.03435



--LEGALH

Train-Test Split

Accuracy: 0.772

Precision: 0.725

Recall: 0.752

F1 score: 0.738

Confusion matrix:

```
[[170  46]
```

```
 [ 40 121]]
```

Cross Validation:

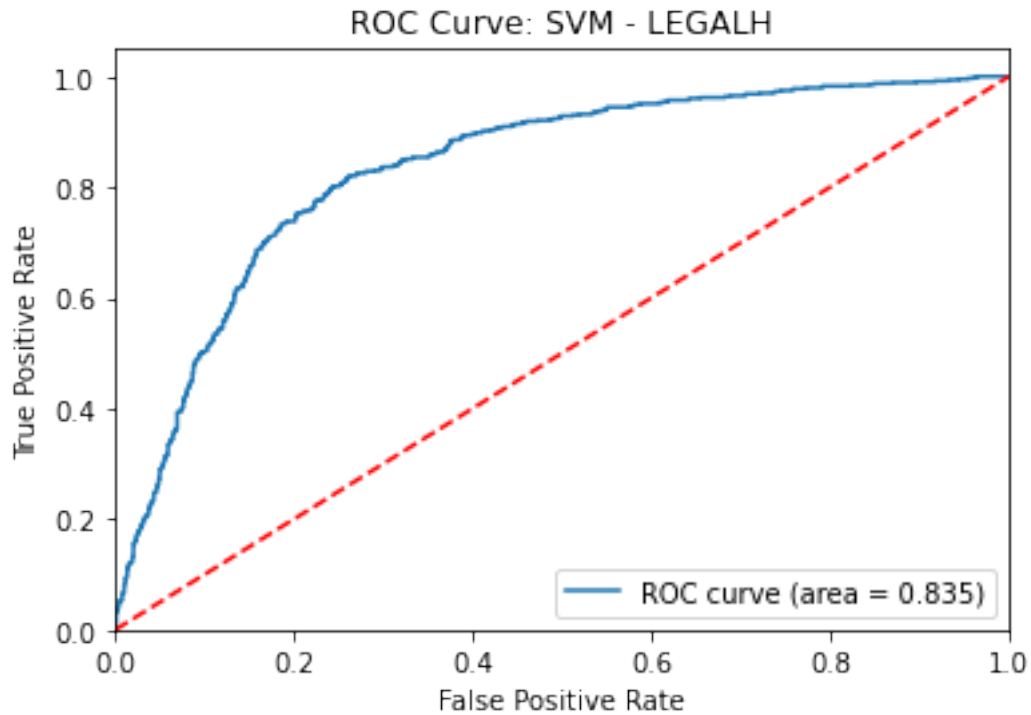
Mean Accuracy: 0.76233 +/- 0.06529

Mean Precision: 0.74130 +/- 0.03971

Mean Recall: 0.72698 +/- 0.03037

Mean F1 score: 0.70 +/- 0.11

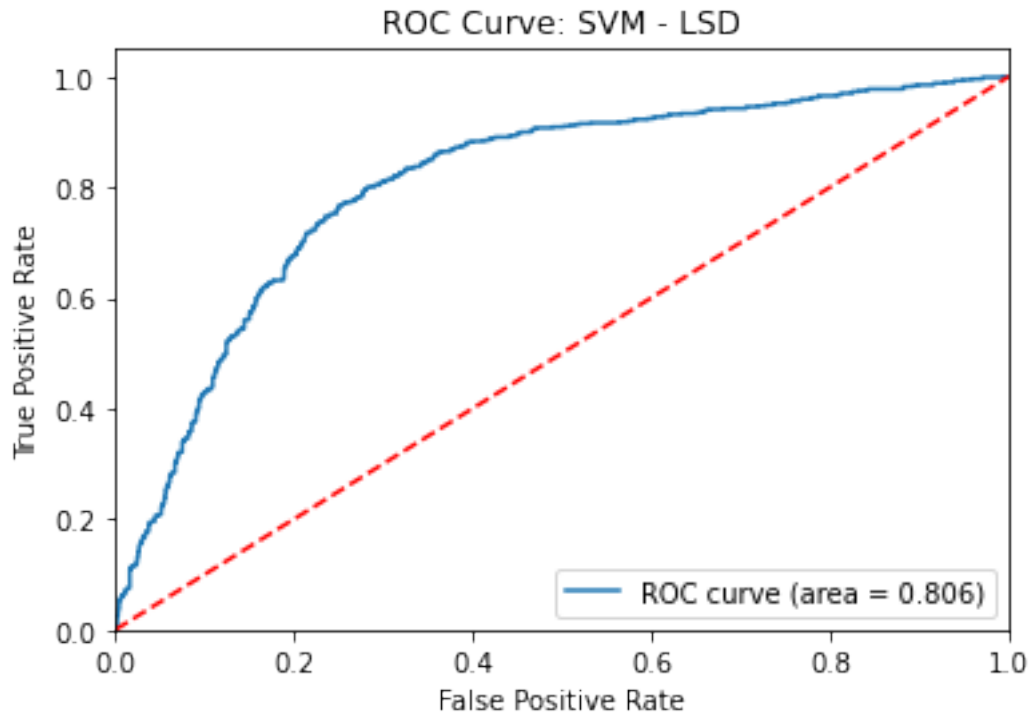
Mean AUC Score: 0.83548 +/- 0.01653



--LSD

Train-Test Split  
Accuracy: 0.719  
Precision: 0.657  
Recall: 0.698  
F1 score: 0.677  
Confusion matrix:  
[[160 58]  
 [ 48 111]]

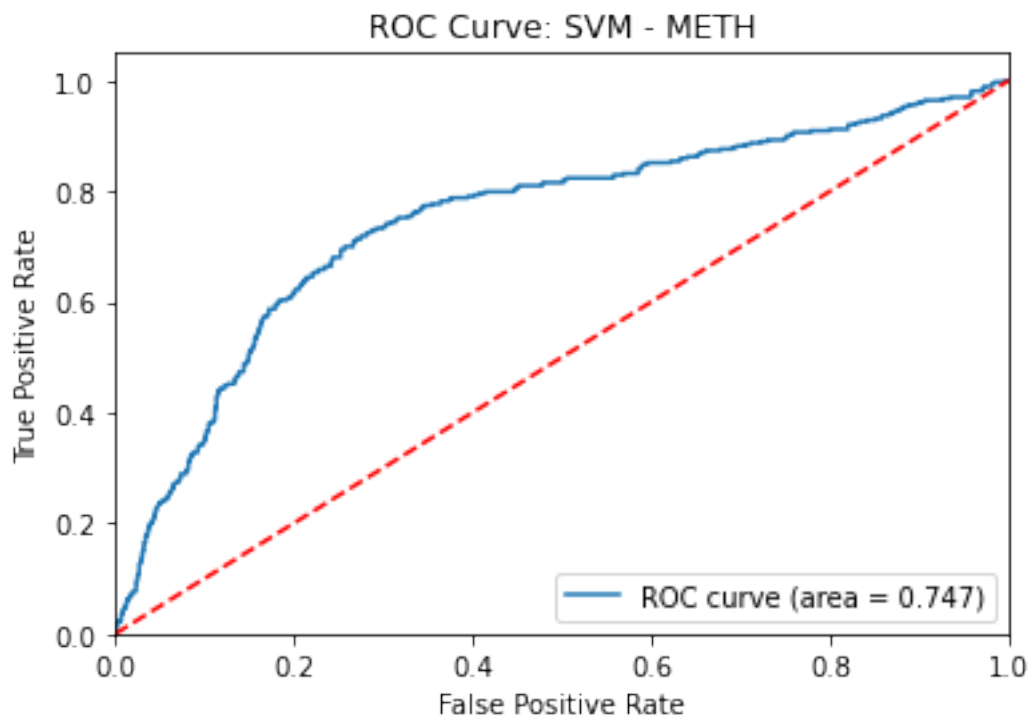
Cross Validation:  
Mean Accuracy: 0.73793 +/- 0.05388  
Mean Precision: 0.71576 +/- 0.02421  
Mean Recall: 0.72917 +/- 0.02731  
Mean F1 score: 0.68 +/- 0.12  
Mean AUC Score: 0.80720 +/- 0.01506



--METH

Train-Test Split  
Accuracy: 0.775  
Precision: 0.580  
Recall: 0.312  
F1 score: 0.406  
Confusion matrix:  
[[263 21]  
 [ 64 29]]

Cross Validation:  
Mean Accuracy: 0.75332 +/- 0.02367  
Mean Precision: 0.55100 +/- 0.03000  
Mean Recall: 0.28094 +/- 0.04308  
Mean F1 score: 0.32 +/- 0.15  
Mean AUC Score: 0.74799 +/- 0.01744



--MUSHROOMS

Train-Test Split

Accuracy: 0.706

Precision: 0.705

Recall: 0.694

F1 score: 0.699

Confusion matrix:

```
[[137  54]
```

```
 [ 57 129]]
```

Cross Validation:

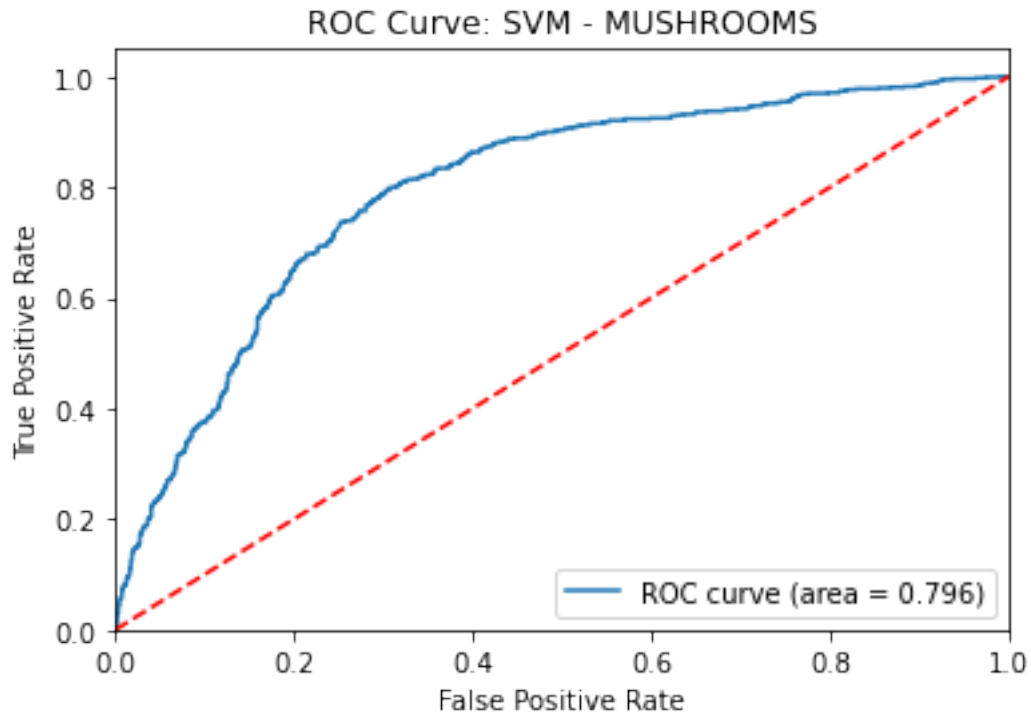
Mean Accuracy: 0.72202 +/- 0.05865

Mean Precision: 0.71737 +/- 0.02675

Mean Recall: 0.73784 +/- 0.03462

Mean F1 score: 0.70 +/- 0.10

Mean AUC Score: 0.79640 +/- 0.01736



--NICOTINE

Train-Test Split

Accuracy: 0.753

Precision: 0.762

Recall: 0.983

F1 score: 0.858

Confusion matrix:

```
[[ 3 88]
```

```
 [ 5 281]]
```

Cross Validation:

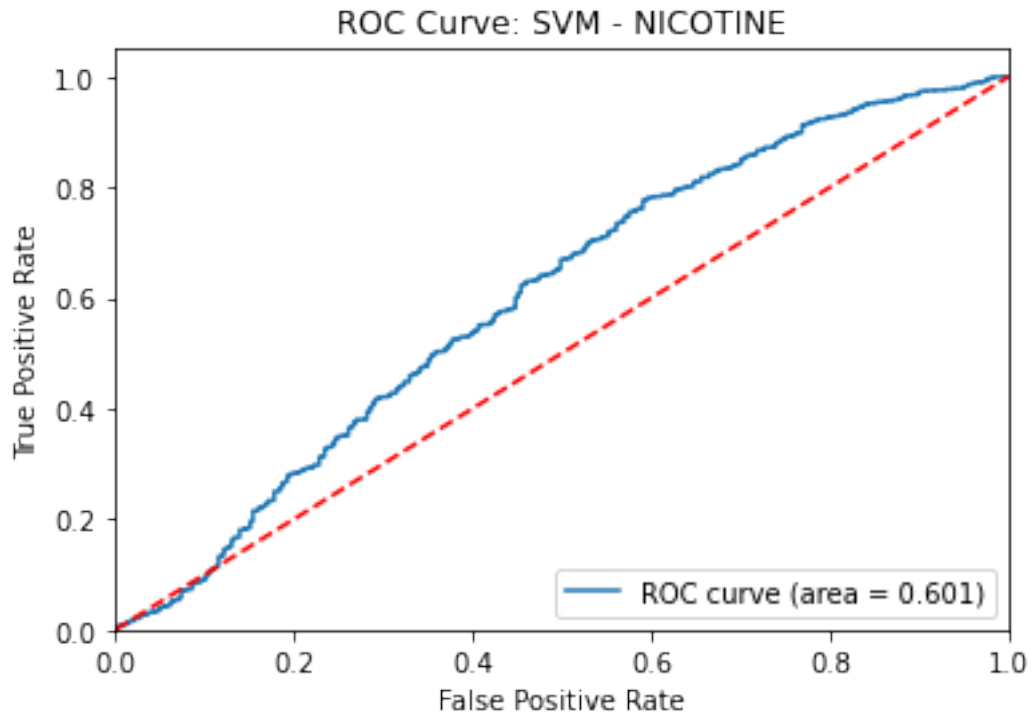
Mean Accuracy: 0.76764 +/- 0.01262

Mean Precision: 0.78076 +/- 0.00277

Mean Recall: 0.99146 +/- 0.00765

Mean F1 score: 0.87 +/- 0.01

Mean AUC Score: 0.59959 +/- 0.03932



## 12.8 Logistic Regression (LR)

[21]:

```
df = pd.read_csv('data6.csv')
print("")

for col_index in range(9, len(df.columns)):

    columnName = df.columns[col_index]
    print()
    print("--" + (columnName))
    print()

    print('Train-Test Split:')

    X = df.iloc[:, 1:9]
    y = df[columnName]
```

```

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
↳random_state=42)

# Create an instance of the model
model_1 = LogisticRegression()

# Train the model on the training data
model_1.fit(X_train, y_train)

# Make predictions on the test data
y_pred1 = model_1.predict(X_test)

# Calculate evaluation metrics
accuracy = accuracy_score(y_test, y_pred1)
precision = precision_score(y_test, y_pred1)
recall = recall_score(y_test, y_pred1)
f1 = f1_score(y_test, y_pred1)
conf_matrix = confusion_matrix(y_test, y_pred1)

# Print the evaluation metrics
print(f"Accuracy: {accuracy:.3f}")
print(f"Precision: {precision:.3f}")
print(f"Recall: {recall:.3f}")
print(f"F1 score: {f1:.3f}")
print("Confusion matrix:")
print(conf_matrix)
print()

N_FOLDS = 5
# Perform cross-validation predictions
y_scores = cross_val_predict(model_1, X_train, y_train, cv=N_FOLDS,
↳method="predict_proba")

fpr, tpr, thresholds = roc_curve(y_train, y_scores[:, 1])
roc_auc = roc_auc_score(y_train, y_scores[:, 1])

print('Cross Validation:')
cv_accuracy = cross_val_score(model_1, X, y, cv=5, scoring='accuracy')
print('Mean Accuracy: %.5f +/- %.5f' % (np.mean(cv_accuracy), np.
↳std(cv_accuracy)))
precision_scores = cross_val_score(model_1, X_train, y_train, cv=N_FOLDS,
↳scoring="precision")

```



```

    print('Mean Precision: %.5f +/- %.5f' % (np.mean(precision_scores), np.
↪std(precision_scores)))
    recall_scores = cross_val_score(model_1, X_train, y_train, cv=N_FOLDS,
↪scoring="recall")
    cv_f1 = cross_val_score(model_1, X, y, cv=5, scoring='f1')
    print('Mean Recall: %.5f +/- %.5f' % (np.mean(recall_scores), np.
↪std(recall_scores)))
    print(f"Mean F1 score: {np.mean(cv_f1):.2f} +/- {np.std(cv_f1):.2f}")
    auc_scores = cross_val_score(model_1, X_train, y_train, cv=N_FOLDS,
↪scoring="roc_auc")
    print('Mean AUC Score: %.5f +/- %.5f' % (np.mean(auc_scores), np.
↪std(auc_scores)))
    print()
    print()

    # Plot ROC curve
    plt.figure()
    plt.plot(fpr, tpr, label='ROC curve (area = %0.3f)' % roc_auc)
    plt.plot([0, 1], [0, 1], 'r--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    # Model
    plt.title('ROC Curve: LR - ' + (columnName))
    plt.legend(loc="lower right")
    plt.show()

```

--AMPHET

Train-Test Split:

Accuracy: 0.671

Precision: 0.669

Recall: 0.624

F1 score: 0.646

Confusion matrix:

```
[[140  56]
```

```
 [ 68 113]]
```

Cross Validation:

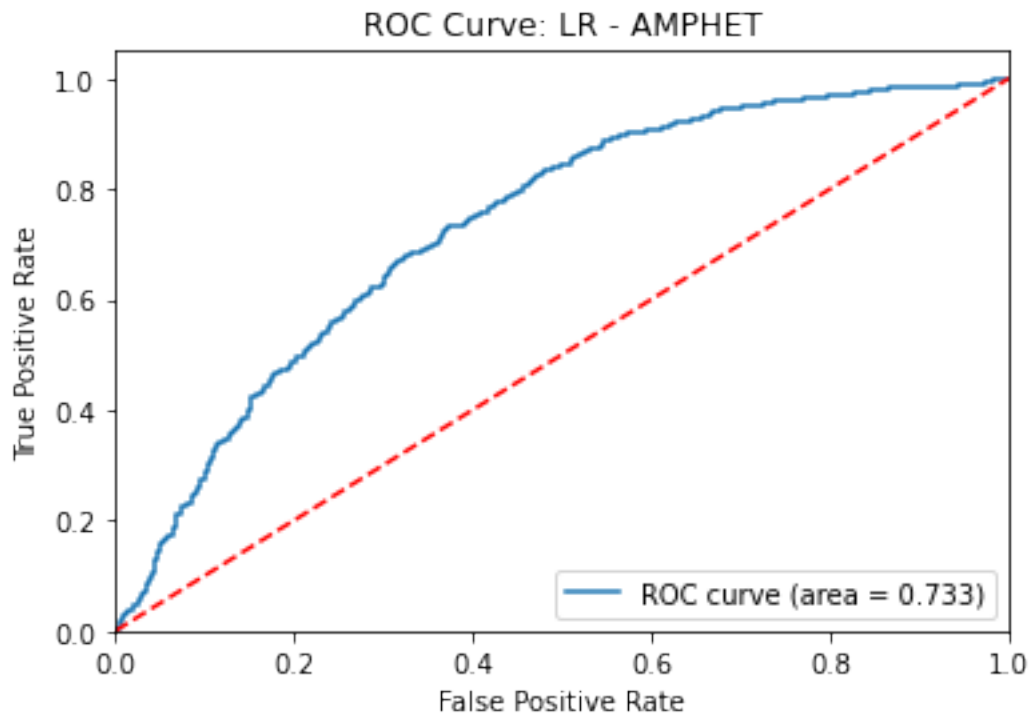
Mean Accuracy: 0.66366 +/- 0.03975

Mean Precision: 0.66113 +/- 0.01859

Mean Recall: 0.62773 +/- 0.04276

Mean F1 score: 0.62 +/- 0.12

Mean AUC Score: 0.73343 +/- 0.01140



--AMYL

Train-Test Split:

Accuracy: 0.729

Precision: 0.667

Recall: 0.241

F1 score: 0.354

Confusion matrix:

[[247 14]

[ 88 28]]

Cross Validation:

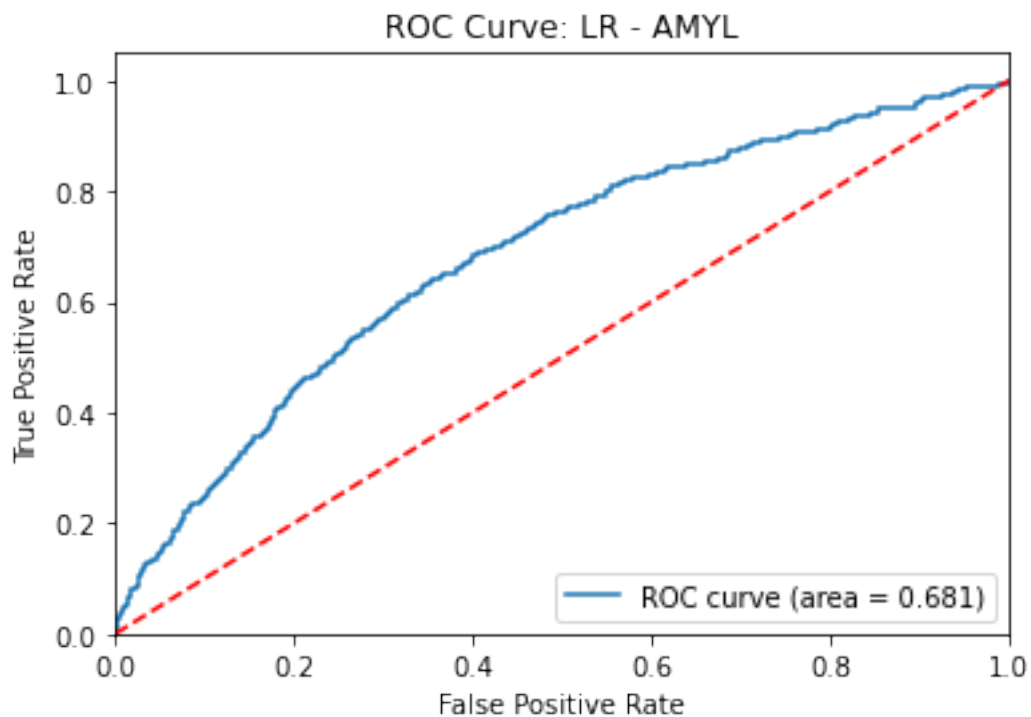
Mean Accuracy: 0.70875 +/- 0.00776

Mean Precision: 0.54613 +/- 0.06122

Mean Recall: 0.17457 +/- 0.02083

Mean F1 score: 0.31 +/- 0.05

Mean AUC Score: 0.68344 +/- 0.04236



--BENZOS

Train-Test Split:

Accuracy: 0.708

Precision: 0.698

Recall: 0.649

F1 score: 0.673

Confusion matrix:

```
[[154  49]
```

```
 [ 61 113]]
```

Cross Validation:

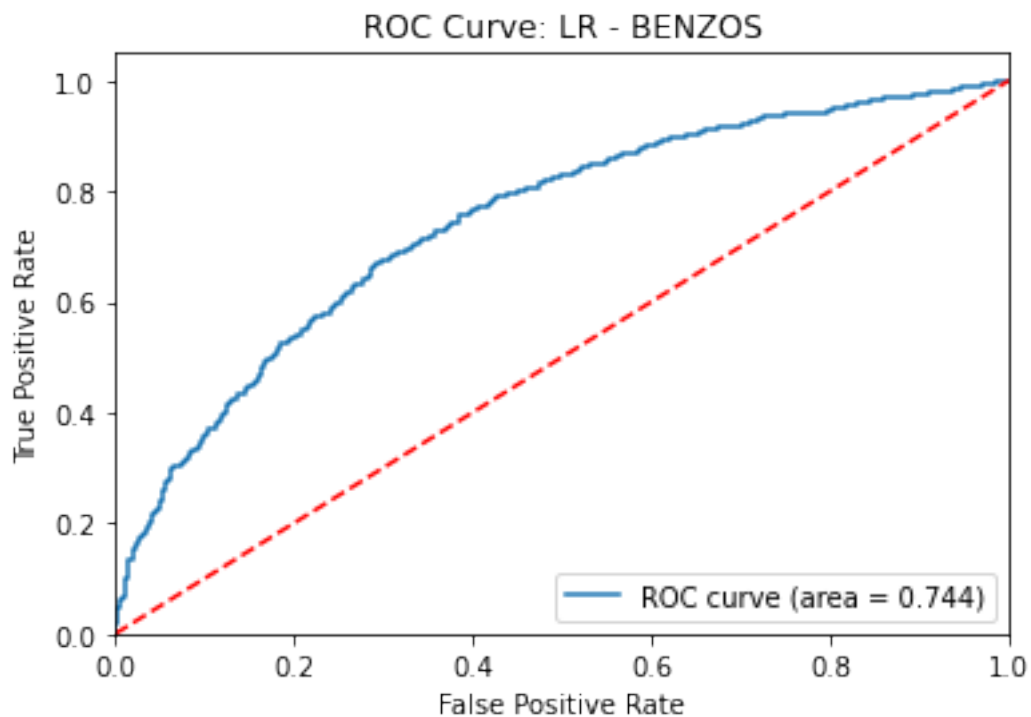
Mean Accuracy: 0.68170 +/- 0.03405

Mean Precision: 0.67785 +/- 0.03934

Mean Recall: 0.63280 +/- 0.05319

Mean F1 score: 0.64 +/- 0.10

Mean AUC Score: 0.74512 +/- 0.02948



--CANNABIS

Train-Test Split:

Accuracy: 0.809

Precision: 0.839

Recall: 0.931

F1 score: 0.883

Confusion matrix:

```
[[ 34  52]
```

```
 [ 20 271]]
```

Cross Validation:

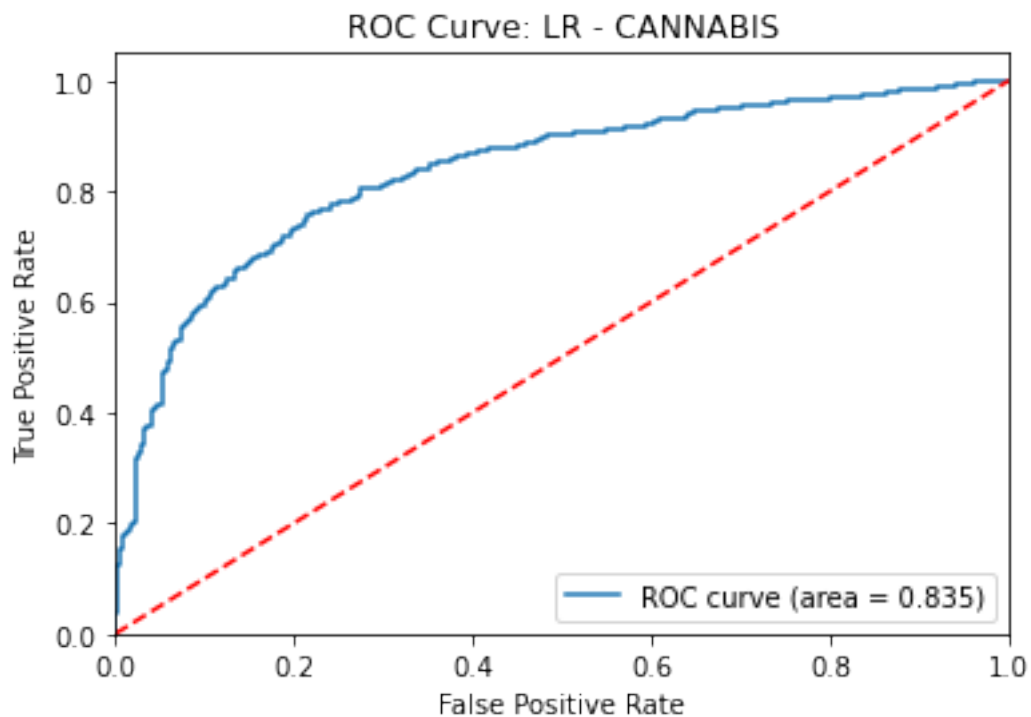
Mean Accuracy: 0.80743 +/- 0.03155

Mean Precision: 0.84378 +/- 0.00787

Mean Recall: 0.93224 +/- 0.01591

Mean F1 score: 0.88 +/- 0.03

Mean AUC Score: 0.83615 +/- 0.02549



--COKE

Train-Test Split:

Accuracy: 0.650

Precision: 0.570

Recall: 0.562

F1 score: 0.566

Confusion matrix:

```
[[159  65]
```

```
 [ 67  86]]
```

Cross Validation:

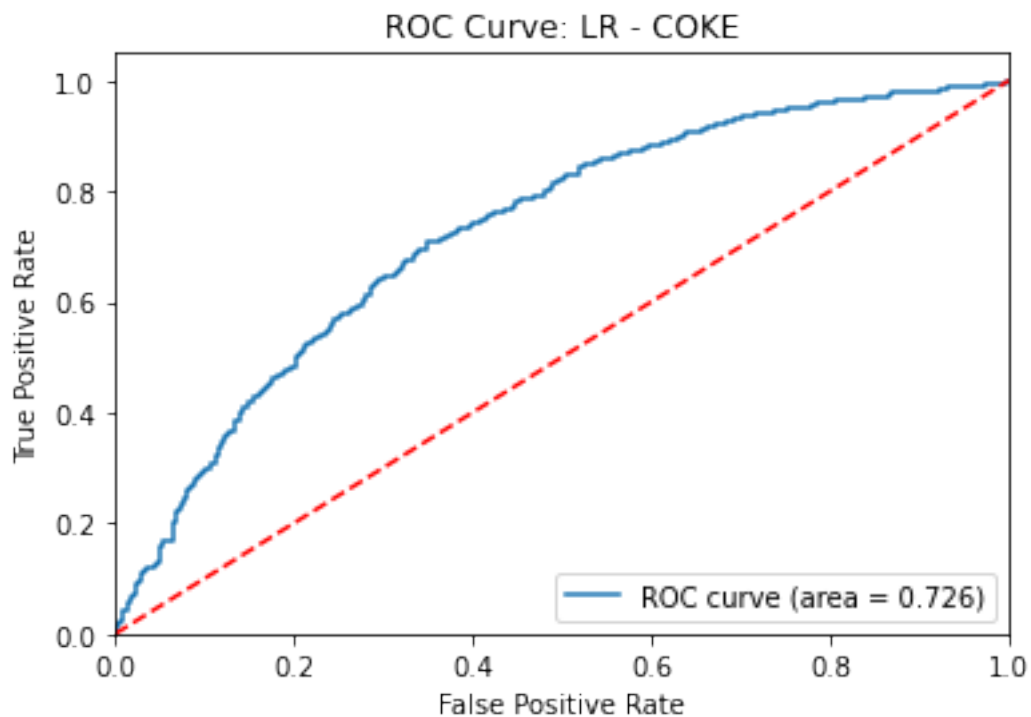
Mean Accuracy: 0.65729 +/- 0.03771

Mean Precision: 0.64825 +/- 0.00938

Mean Recall: 0.60084 +/- 0.04603

Mean F1 score: 0.59 +/- 0.10

Mean AUC Score: 0.72773 +/- 0.01924



--CRACK

Train-Test Split:

Accuracy: 0.886

Precision: 0.500

Recall: 0.047

F1 score: 0.085

Confusion matrix:

```
[[332  2]
```

```
 [ 41  2]]
```

Cross Validation:

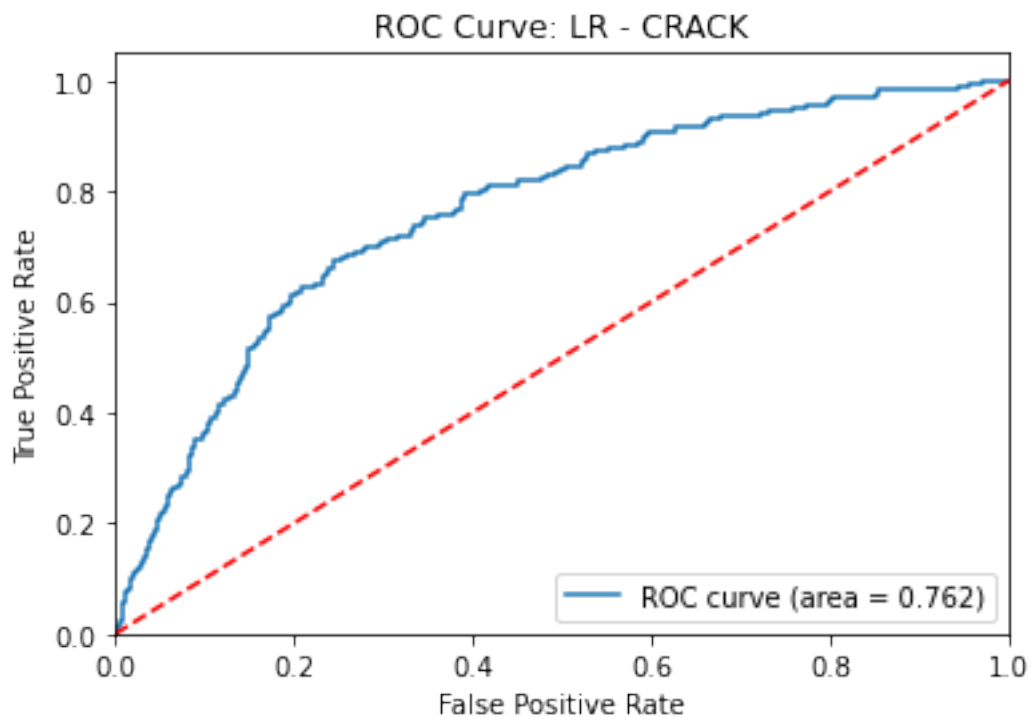
Mean Accuracy: 0.85942 +/- 0.00964

Mean Precision: 0.55000 +/- 0.25739

Mean Recall: 0.06977 +/- 0.03289

Mean F1 score: 0.07 +/- 0.05

Mean AUC Score: 0.76496 +/- 0.04602



--ECSTASY

Train-Test Split:

Accuracy: 0.687

Precision: 0.665

Recall: 0.609

F1 score: 0.636

Confusion matrix:

```
[[156  52]
```

```
 [ 66 103]]
```

Cross Validation:

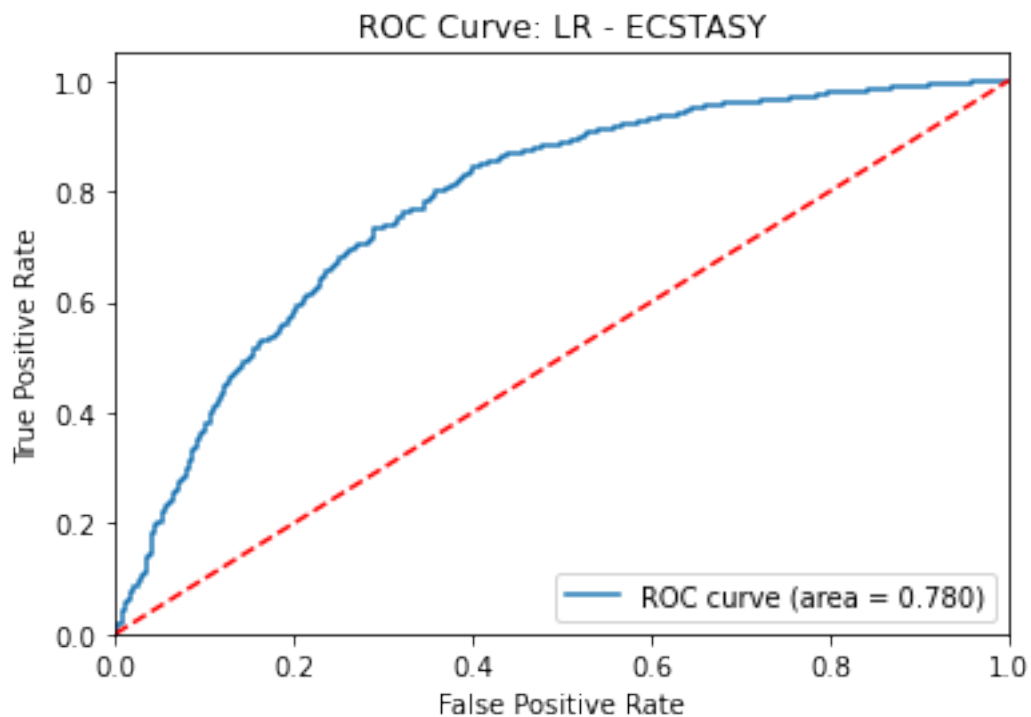
Mean Accuracy: 0.69072 +/- 0.05636

Mean Precision: 0.70353 +/- 0.03469

Mean Recall: 0.65468 +/- 0.03611

Mean F1 score: 0.62 +/- 0.16

Mean AUC Score: 0.78173 +/- 0.01923



--HEROIN

Train-Test Split:

Accuracy: 0.859

Precision: 0.500

Recall: 0.075

F1 score: 0.131

Confusion matrix:

```
[[320  4]
```

```
 [ 49  4]]
```

Cross Validation:

Mean Accuracy: 0.84934 +/- 0.00829

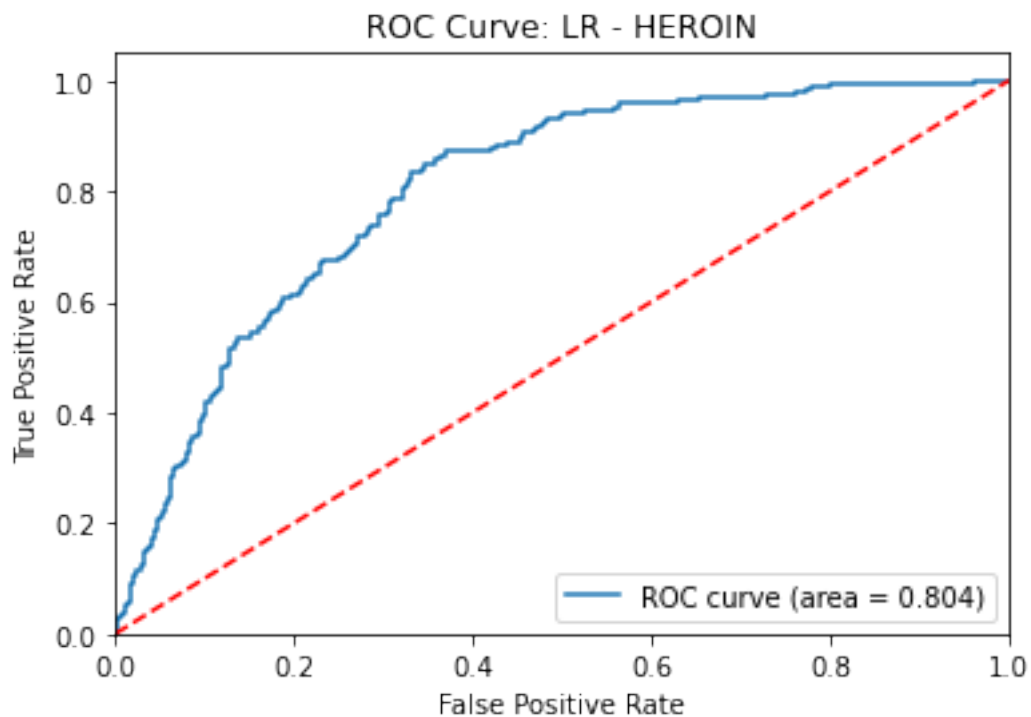
Mean Precision: 0.44137 +/- 0.12906

Mean Recall: 0.11478 +/- 0.05721

Mean F1 score: 0.17 +/- 0.05

Mean AUC Score: 0.80353 +/- 0.03509





--KETAMINE

Train-Test Split:

Accuracy: 0.828

Precision: 0.700

Recall: 0.101

F1 score: 0.177

Confusion matrix:

```
[[305  3]
```

```
 [ 62  7]]
```

Cross Validation:

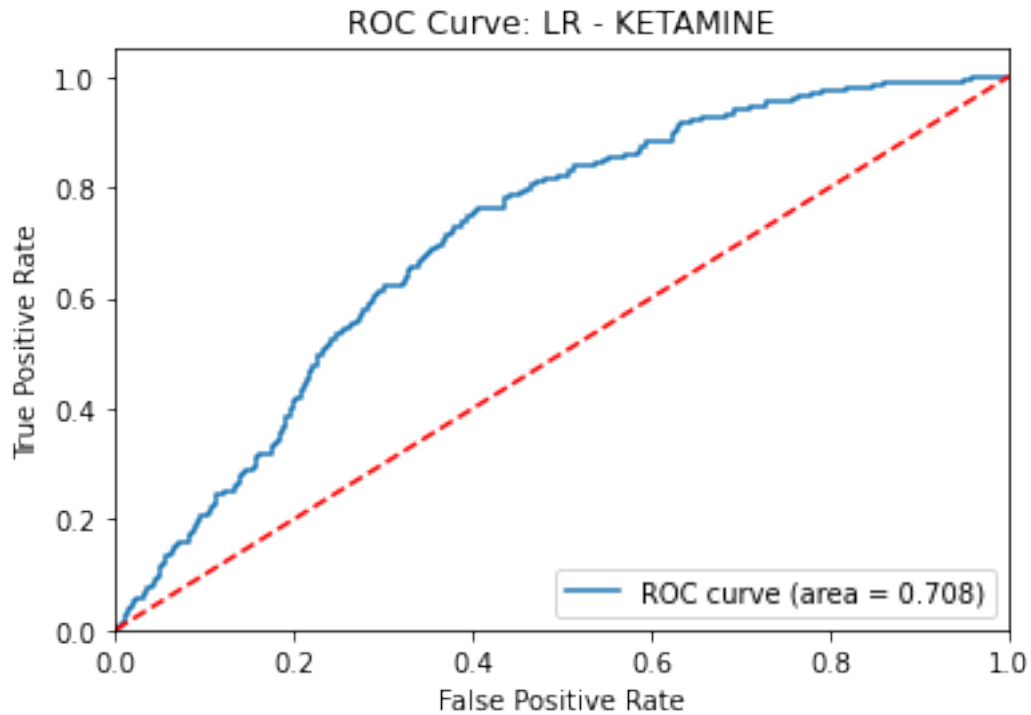
Mean Accuracy: 0.78090 +/- 0.01996

Mean Precision: 0.39416 +/- 0.09538

Mean Recall: 0.05221 +/- 0.01857

Mean F1 score: 0.12 +/- 0.06

Mean AUC Score: 0.70930 +/- 0.02357



--LEGALH

Train-Test Split:

Accuracy: 0.790

Precision: 0.770

Recall: 0.727

F1 score: 0.748

Confusion matrix:

```
[[181  35]
```

```
 [ 44 117]]
```

Cross Validation:

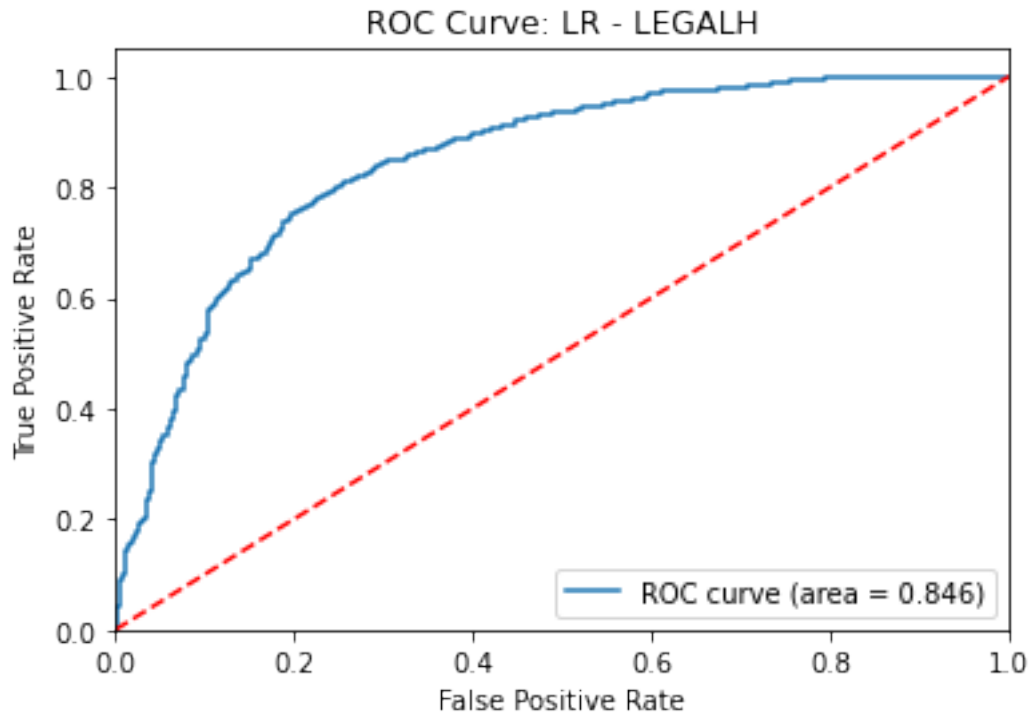
Mean Accuracy: 0.76711 +/- 0.07112

Mean Precision: 0.74635 +/- 0.05133

Mean Recall: 0.70000 +/- 0.04121

Mean F1 score: 0.69 +/- 0.15

Mean AUC Score: 0.84653 +/- 0.01829



--LSD

Train-Test Split:

Accuracy: 0.711

Precision: 0.667

Recall: 0.629

F1 score: 0.647

Confusion matrix:

```
[[168  50]
```

```
 [ 59 100]]
```

Cross Validation:

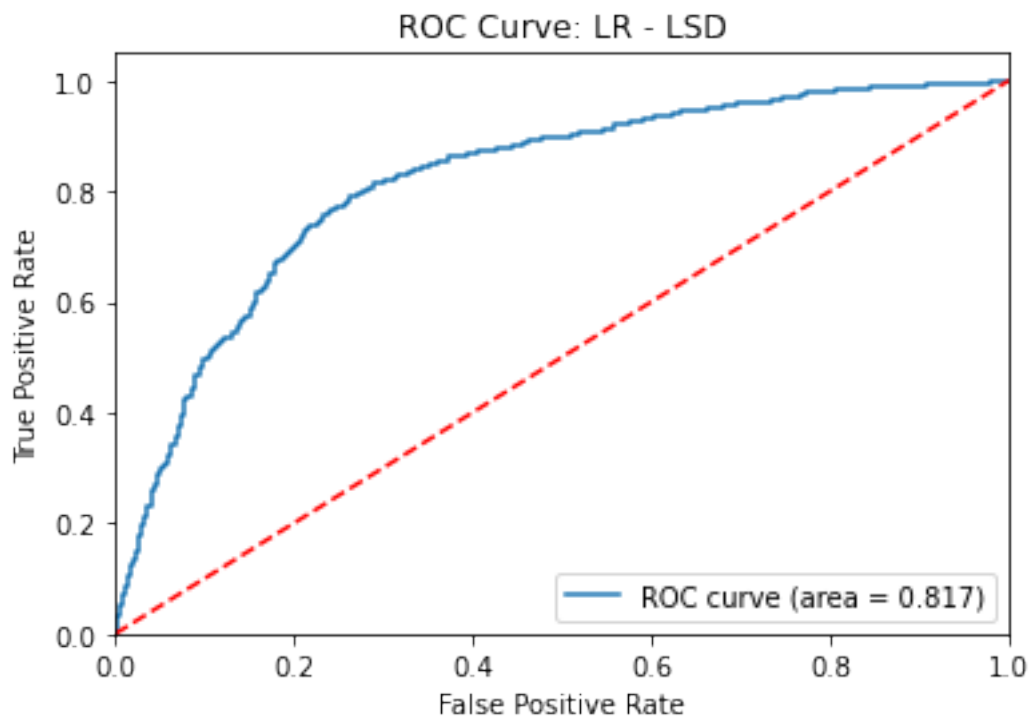
Mean Accuracy: 0.73740 +/- 0.05233

Mean Precision: 0.73105 +/- 0.02569

Mean Recall: 0.68804 +/- 0.02853

Mean F1 score: 0.67 +/- 0.13

Mean AUC Score: 0.81734 +/- 0.01434



--METH

Train-Test Split:

Accuracy: 0.777

Precision: 0.588

Recall: 0.323

F1 score: 0.417

Confusion matrix:

```
[[263  21]
```

```
 [ 63  30]]
```

Cross Validation:

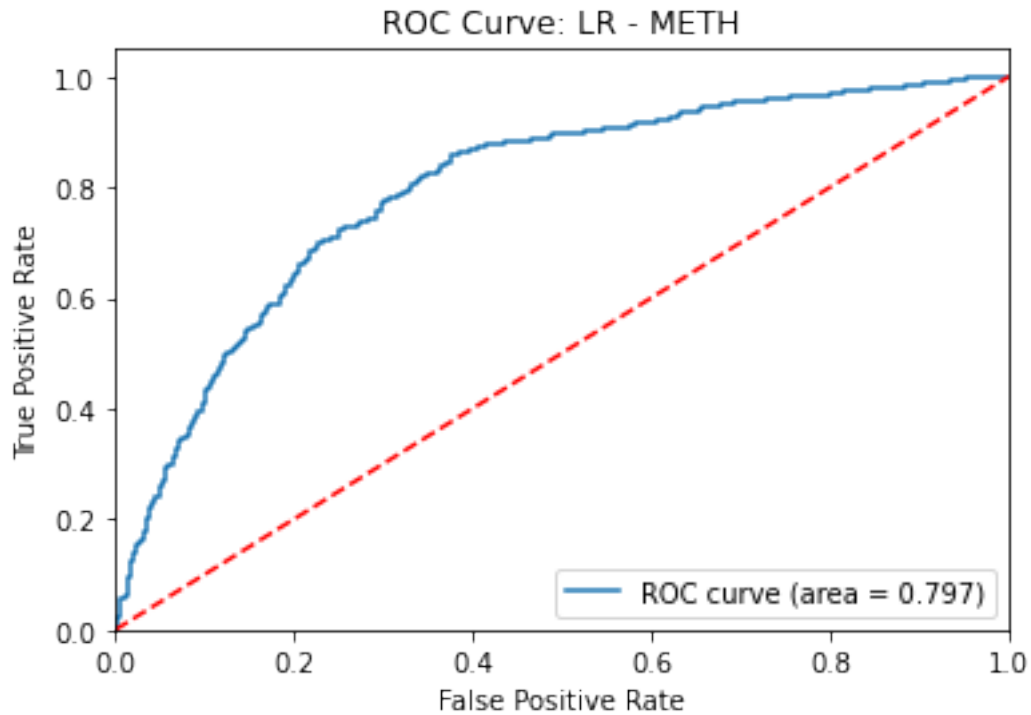
Mean Accuracy: 0.76817 +/- 0.02152

Mean Precision: 0.57800 +/- 0.03203

Mean Recall: 0.34718 +/- 0.03161

Mean F1 score: 0.40 +/- 0.15

Mean AUC Score: 0.79748 +/- 0.01165



--MUSHROOMS

Train-Test Split:

Accuracy: 0.692

Precision: 0.708

Recall: 0.640

F1 score: 0.672

Confusion matrix:

```
[[142  49]
```

```
 [ 67 119]]
```

Cross Validation:

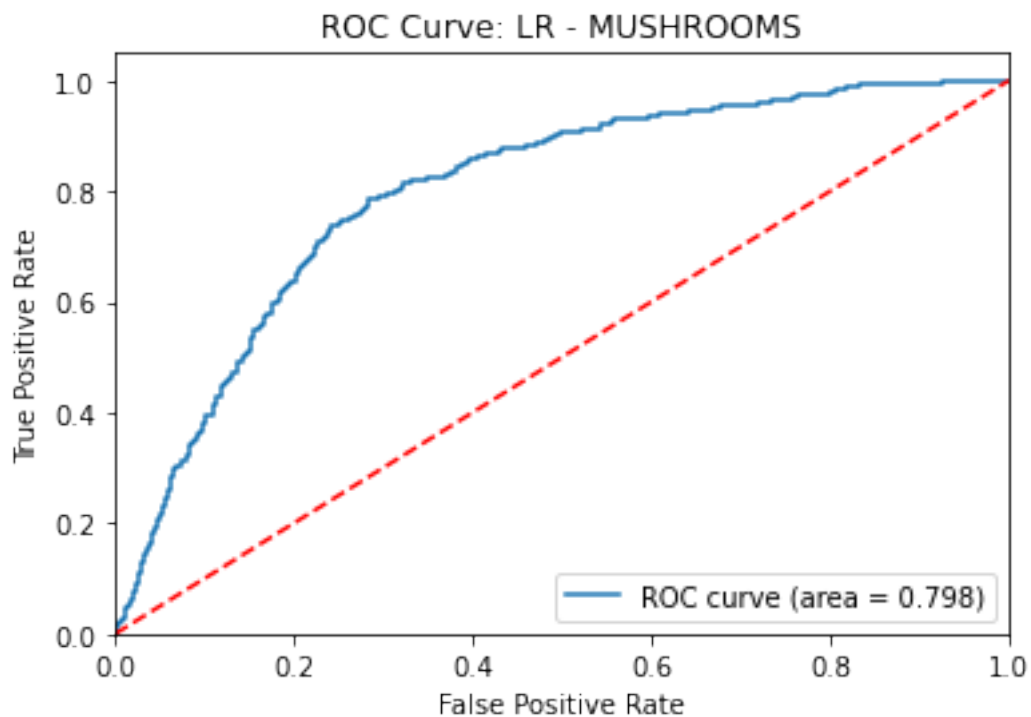
Mean Accuracy: 0.71406 +/- 0.06620

Mean Precision: 0.73707 +/- 0.02962

Mean Recall: 0.70858 +/- 0.04124

Mean F1 score: 0.67 +/- 0.15

Mean AUC Score: 0.79984 +/- 0.02449



--NICOTINE

Train-Test Split:

Accuracy: 0.759

Precision: 0.770

Recall: 0.972

F1 score: 0.859

Confusion matrix:

```
[[ 8 83]
```

```
 [ 8 278]]
```

Cross Validation:

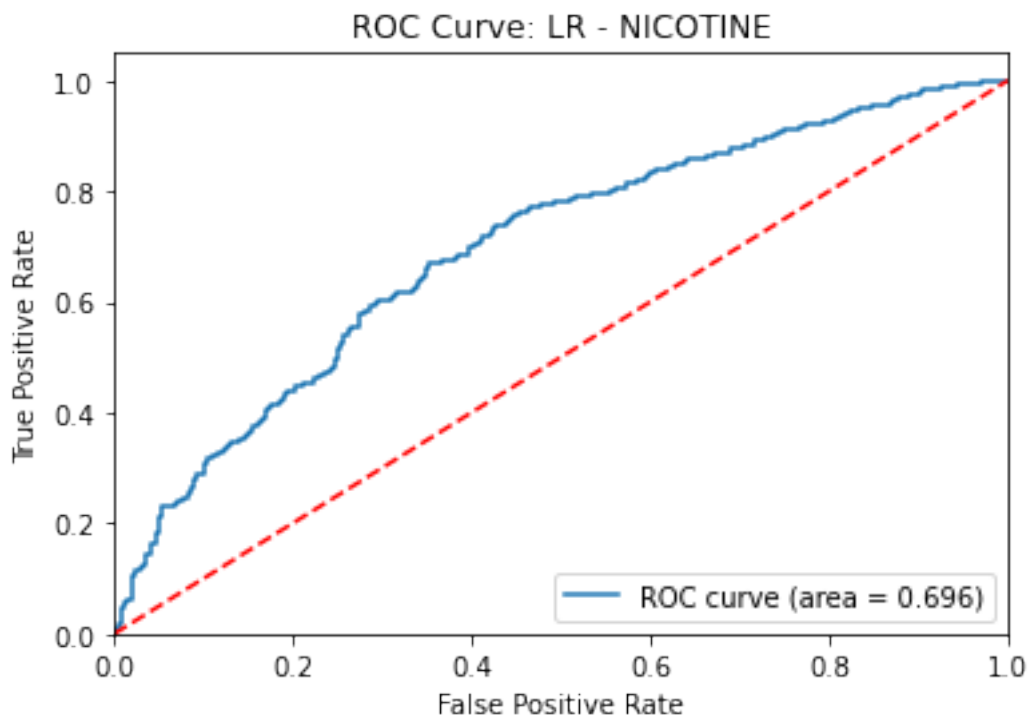
Mean Accuracy: 0.77029 +/- 0.02066

Mean Precision: 0.78984 +/- 0.00339

Mean Recall: 0.97865 +/- 0.00855

Mean F1 score: 0.87 +/- 0.02

Mean AUC Score: 0.69687 +/- 0.02093



## 13 Results & Discussion

After training and testing the different classification models, there are models that have high percentages for their accuracy, precision, recall, and F1 Score for specific drugs. However, in the context of this research, it is important to note that these percentages on their own do not necessarily give us the entire picture of the effectiveness of these models. There were many models that had recall percentages well above 90%, however, for this paper, we are also taking into account the AUC score and the F1 score because it gives us a greater understanding of the model's ability to discriminate between classes and its true positive and false positive rates, rather than predicting the majority of outputs.

For example, there were many drugs under the SVM model that scored relatively high for a specific drug's accuracy, precision, recall, and F1 Score. However, the AUC score was a little over 0.5, which is essentially random chance. Since the AUC score was not near 1 for some of these results even with the high accuracy, we cannot use these results because it demonstrates that the dataset was imbalanced for that specific drug (such as a very hard-core, uncommon drug with very few users) where one class significantly outnumbered the other. For such hard drugs, the model can accurately predict that most people were non-users, however, its AUC score tells us that it lacks the ability to discriminate between users and non-users. The classifier may have learned to predict the majority class more frequently, which explains the high accuracy but the poor performance in distinguishing the minority class. As a result, for this paper, we finalized our results with models that had both relatively high F1 and AUC scores for specific drugs with a model.

For this research, we purposefully also calculated the AUC and F1 scores of each drug with each

model. Together, by analyzing the AUC score along side the model's F1 Score for each drug allowed us to conclude the most effective model that accurately predicts true drug-use and has the ability to distinguish across various thresholds.

The following are the results from the study. The table showcases the models with relatively high F1 and AUC scores for specific drugs across all models.

Model	Drug Name	F1 Score	AUC Score
Linear Discriminant Analysis	Cannabis	0.87	0.835
Linear Discriminant Analysis	LegalH	0.70	0.846
K-Nearest Neighbors	Cannabis	0.86	0.771
Gaussian Naive Bayes	Cannabis	0.82	0.834
Gaussian Naive Bayes	LegalH	0.7	0.8377
Random Forest	Cannabis	0.87	0.815
Random Forest	Coke	0.61	0.837
Random Forest	LSD	0.67	0.795
Random Forest	Mushrooms	0.70	0.792
Gradient Boosting Machine	Cannabis	0.87	0.818
Gradient Boosting Machine	Ecstasy	0.67	0.785
Gradient Boosting Machine	LegalH	0.68	0.809
Support Vector Machine	Cannabis	0.88	0.781
Support Vector Machine	Ecstasy	0.68	0.782
Support Vector Machine	LegalH	0.70	0.835
Support Vector Machine	Mushrooms	0.70	0.796
Logistic Regression	Cannabis	0.88	0.836
Logistic Regression	Ecstasy	0.62	0.781
Logistic Regression	LegalH	0.69	0.846
Logistic Regression	LSD	0.67	0.817
Logistic Regression	Mushrooms	0.67	0.799
Logistic Regression	Nicotine	0.87	0.696

The following is a condensed version of the above list, showing models that have great potential for future applications of our research results to increase the effectiveness of these models, and using other models for these specific drugs.

Model	Drug Name	F1 Score	AUC Score
Logistic Regression	Cannabis	0.88	0.836
Linear Discriminant Analysis	LegalH	0.70	0.846
Logistic Regression	Nicotine	0.87	0.696

## 14 Conclusion & Future Plans

In summary, our study demonstrates that machine learning algorithms are capable of accurately predicting the use of drugs for an individual using their data related to demographics and personality factors. Specifically, the ability for these models to predict the use of Cannabis, LegalH, and Nicotine was effective. Based on these results, the classification model we identified to be the most



effective for predicting the use of the above drugs, given high F1 and AUC scores, was Logistic Regression for the drug Cannabis. Over the next few years, we plan to conduct additional testing using different machine learning training and testing techniques to increase the F1 and AUC scores for these drugs and other un-tested drugs using different drug-use data repositories. Following these additional tests, the next step is to create a website and an IOS software app to ensure the widespread accessibility of the results of this study for individuals, families, and counselors. The results of this study can be used as an additional tool in assessing the risk of drug use for specific drugs in adolescents to counteract the drug abuse epidemic in America.

## 15 References

1. Black, J. (2022). An Introduction to Machine Learning for Classification and Prediction. Family Practice, XX, 1–5.
2. Federal Bureau of Prisons. (2023, November 11). BOP statistics: Inmate offenses. Federal Bureau of Prisons; Federal Bureau of Prisons. [https://www.bop.gov/about/statistics/statistics\\_inmate\\_offenses.jsp](https://www.bop.gov/about/statistics/statistics_inmate_offenses.jsp)
3. Johns Hopkins Medicine. (2019). Substance Abuse / Chemical Dependency. John Hopkins Medicine. <https://www.hopkinsmedicine.org/health/conditions-and-diseases/substance-abuse-chemical-dependency>
4. National Center for Drug Abuse Statistics. (2020). NCDAS: Substance Abuse and Addiction Statistics [2020]. National Center for Drug Abuse Statistics. <https://drugabusestatistics.org/>
5. National Drug Intelligence Center. (2006, January). The Impact of Drugs on Society - National Drug Threat Assessment 2006. Justice.gov. <https://www.justice.gov/archive/ndic/pubs11/18862/impact.htm>
6. Substance Use Has Risen During COVID-19 Pandemic. (2022, March 15). Wwww.rpc.senate.gov. <https://www.rpc.senate.gov/policy-papers/substance-use-has-risen-during-covid-19-pandemic>
7. UCI Machine Learning Repository. (2016). Archive.ics.uci.edu. <https://archive.ics.uci.edu/dataset/373/drug+consumption+quantified>

[ ]: