CSC 859: AI Ethics and Explainability
Fall 2022

San Francisco State University

Team Project: Development and Ethics Audit of AI Application
Team 03

Final Report

Lawrence Ng (lng3@mail.sfsu.edu)
Darshil Dhameliya (ddhameliya@mail.sfsu.edu)
Warren Singh (wsingh@mail.sfsu.edu)

December 10, 2022

Contents

1 Executive Summary

Drug use is a societal problem with impacts on communities at all levels (economic, political, logistic, medical, social, etc.) with knock-on effects at the municipal, state, and federal levels. While drug use or dependency may depend on a multitude of factors, particular personality measures such as the 'big five' have the most evidence for accuracy, reliability, and robustness. This paper attempts to use these personality measures along with demographic information as predictive factors for identifying at-risk individuals for drug usage, and specifically whether certain personality traits are associated with usage of high risk drugs such as crack, cocaine, or heroin.

We also approach our machine learning experiments in adherence with AI Ethics best practices. We perform audits on the dataset chosen and the experiments as a whole, making sure to address issues through explainability and objective interpretation of the accuracy measures of our models. Our results and analyses show that our models might not generalize well to produce reliable predictions, and we provide recommendations on future steps based on our audit results on how the model can be improved for such use cases.

2 Introduction

The purpose of this report is to describe the results of our team's attempt at creating a machine learning model using random forest that can predict at-risk drug use given demographic and personality measures for a given individual. In particular, we want to develop the machine learning model by following best AI ethical practices by auditing our data, model, and results. Successful development of a model under these conditions would enable us to provide a trustworthy, accurate, and explainable prediction system.

In this section, we provide the motivation for why we chose to develop an AI application that can predict at-risk drug use and the goals of the overall project. In section 3, we describe the dataset used for developing the machine learning model and perform a data audit in adherence to best practices. In Section 4, we detail the setup of our experiments and their results. In section 5, we perform an ethics audit on our entire AI development process. In section 6, we summarize our final results and provide our conclusion and recommendation.

2.1 Motivation

Given the vast impact of drug use on our society, our team is interested in finding out the factors that can accurately predict an individual's risk level with regards to harmful drug use. Various researchers have postulated that there is a relationship between drug use and personality traits. [1] [2] So we hope to answer the following question: are there personality measures which predict at-risk drug use, and in particular, is it possible to use an explainable AI system (i.e., Random Forest) to capture this predictive relationship? If so, at-risk individuals/groups could be identified earlier in the addiction process, and resources can be more efficiently and optimally allocated to ameliorate or prevent harms. This could lead to lower healthcare costs, lower incarceration rates, and lower crime rates among other positive outcomes.

2.2 Problem Description

The problems that we look to answer in this project include:

1. Given demographics data and results of personality tests, can we predict with confidence whether an individual is likely to consume drugs or not?
2. Can we narrow down which drug the individual is likely to experiment with?
3. More generally, are there personality factors which generalize to predict at-risk behavior involving drugs?

2.3 Case Study Goals

In this report, we aim to achieve the following objectives:

1. Validate the result of the research paper accompanying the dataset that we chose.
2. Find predictive relationship between personality measures and at-risk drug behavior with different substance subtypes.
3. Explore generalizability of personality factors to at-risk drug usage.
4. Audit and evaluate experiment efforts to achieve all previous goals, especially keeping in mind general ethical principles.

3 Data Description and Analysis

We identified a dataset from UC Irvine's Machine Learning Repository as containing characteristics that we see fit in tackling the problem. [3]

The dataset is originally tabular with features as described in the next subsection. It contains 1885 samples in total with no missing data. The dataset was primarily created by collecting responses from an online survey. The participants were asked to provide information regarding their use of specific illicit and non-illicit drugs. Based on the responses, each participant is classified for each drug using CL0 to CL6 labels in the original dataset, which can be seen as the ground truth in the dataset.

The features 'Gender,' 'Country,' and 'Ethnicity' are categorical, and these categories are originally described using different numbers on a normalized scale. 'Age' and 'Level of Education' are labeled using ranges of values converted to normalized numerical scale. All other features are numerical on a normalized scale. More detailed description and analysis of the data is available in [4]

3.1 Data Attributes

This drug consumption dataset contains a total of 1885 samples. Each sample represents an individual for whom the following attributes are known:

1. Age

2. Gender
3. Level of Education
4. Country of current residence
5. Ethnicity
6. BIS-11 personality measurement for impulsivity (Impulsive)
7. ImpSS personality measurement for sensation seeking (SS)
8. NEO-FFI-R personality measurements for
    (a) Neuroticism (Nscore)
    (b) Extraversion (Escore)
    (c) Openness to experience (Oscore)
    (d) Agreeableness (Ascore)
    (e) Conscientiousness (Cscore)

The ground truth in the original dataset uses the definition described below: (Self-reported use of 18 legal and illegal drugs based on 7 categories)

- Never used: CL0
- Used over a decade ago: CL1
- Used within last decade: CL2
- Used within last year: CL3
- Used within last month: CL4
- Used within last week: CL5
- Used within last day: CL6

This is the original set of drugs asked about in the survey:

1. Alcohol
2. Amphetamine
3. Amyl Nitrite
4. Benzodiazepine
5. Caffeine
6. Cannabis

7. Chocolate
8. Cocaine
9. Crack
10. Heroin
11. Ketamine
12. Legal Highs
13. LSD
14. Methadone
15. Magic Mushrooms
16. Nicotine
17. Semeron (a fictitious drug used as a filtering mechanism for "overclaimers")
18. Volatile Substance (VSA)

3.2 Data Audit

Prior to conducting any machine learning training, we first audit the database using the checklist from Prof. Petkovic [5], which seeks to answer the following questions:

How are the data obtained?

The data is obtained via online surveys where the personality tests are conducted and demographics and self-reported drug usage is obtained. It should be noted that personality scores may be biased as respondents may be seeking to provide more normative responses, simply due to the self-reported nature of the data collection mechanism. Drug usage might also not be perfectly reliable due to self-report bias and error, though generally if respondents are acting in honest, good-faith responses, the inaccuracies may only be slight (e.g., stating they used a substance in the last month when it was in the last week, or similarly).

How are class labels obtained/verified?

All information is self-reported, thus ground truth is only known to individual respondents. We place our trust on respondents being as accurate as they can be.

Is demography well covered, in an adequate and fair way?

Age Group	Composition
18-24	34.11%
25-34	25.52%
35-44	18.89%
45-54	15.60%
55-64	4.93%
65+	0.95%

Countries	Composition
UK	55.38%
USA	29.55%
Canada	4.62%
Australia	2.86%
Republic of Ireland	1.06%
New Zealand	0.27%
Other	6.26%

Ethnicity	Composition
White	91.25%
Other	3.34%
Black	1.75%
Asian	1.38%
Mixed White/Asian	1.06%
Mixed White/Black	1.06%
Mixed Black/Asian	0.16%

Education Level	Composition
Left school before 16 years old	1.49%
Left school at 16 years old	5.25%
Left school at 17 years old	1.59%
Left school at 18 years old	5.31%
Some college or university, no certificate or degree	26.84%
Professional certificate or diploma	14.32%
Bachelors degree or equivalent	25.46%
Masters degree	15.01%
Doctorate degree	4.72%

Tables: Demographic features broken down by corresponding subcategories

We can observe in the tables that there is, for all intents and purposes, an even split of male and female respondents. 60% of respondents are between 18 - 35 and two-thirds have finished some college or university or beyond. The vast majority of respondents are white and from English-speaking countries (with around 80% from the UK and the US combined, and decreasing contributions from other English-speaking countries around the world.) All of this is unsurprising: the younger demographic is more likely to respond to online surveys, and the original paper's authors are based in the UK seeking to understand the relationship between drug usage and personality scores.

Is the data unbalanced?

Illegal behavior tends to be a small minority in general, and drug usage is no exception to this rule. As such, drug abuse forms a minority of cases, especially with respect to highly illegal substances. The original authors use decade-based aggregation for users/non-users to achieve balanced datasets; undersampling to produce balanced experimental sets is also explored as an option.

What are the number and type of features?

There are a total of 12 features, which are made up of demographic information and personality scores. Of the demographic features, gender, country of residence, and ethnicity are categorical features. The rest of demographic and personality score features are numerical. The 19 drug usage labels are also categorical, split into the 7 aforementioned categories.

Are there missing values?

There are no missing values in the dataset.

Are there enough samples compared to the number of features used?

It is best practice to use data with at least 10 times more samples compared to the number of features. Since we have 1885 samples and 12 features, our dataset conforms with best practices.

List and describe features, documented formats.

Both [4] and the [3] gives a complete list of dataset and feature information. [4] provides a graphical breakdown of each class for each feature, allowing for quick analysis of the data.

Are privacy issues addressed?

Data is collected with the consent of respondents for the specific purpose of this experiment. Feature values have been normalized to anonymize respondents' identities so that responses cannot be traced back to specific individuals which helps address privacy concerns.

4 Application and Results of ML Methods

For our project, we used random forest using R following ML best practices on the chosen data after some preprocessing using Microsoft Excel. We outline particular kinds of pre-

processing later in the experiment subsections. We performed the experiments listed below (though not necessarily in the order presented):

- **Experiment 0:** Verify results obtained by the authors of the original paper.
- **Experiment 1:** Create a generalized model that would enable us to categorize an individual as one of three categories with regards to their likelihood to use drugs: no concern, slight concern, or very concerning.
- **Experiment 2:** Create different classification models for particular drugs of interest to better predict an individual's likelihood of using a given drug.
- **Experiment 3:** Create different classification models for particular drugs using the specific features that produced the most optimal machine learning models in the original paper to see if results replicate or improve.

Our attempt to verify results obtained by the original authors in Experiment 0 were hampered by the reasons outlined below:

1. The authors used a variety of methods to find the most accurate machine learning algorithm. Their results indicate that decision trees produced the most accurate result for most drugs, but our attempt at creating a machine learning model uses only random forest. The two methods are not perfectly comparable.
2. The authors ran through 166 million decision tree models per drug to find the optimal model. We are unable to dedicate the same effort.
3. They did not provide data and results for their random forest models, so we cannot compare if our own produced results would have indeed matched up with their random forest models.

Due to the above reasons, we moved onto our first experiment.

As an additional note, the original paper implicitly makes use of the Copernican principle as applied to statistics, and we have done the same in the attempt to 'forecast' problematic drug usage.

In short, the Copernican principle with uninformed priors (as a special case of an application of a Bayesian analysis) assumes there is nothing special about a particular moment in time being observed. Therefore, for our experiment, (and in general with uninformed priors) one may assume that the length of time since the last usage of a drug will roughly be the amount until the next usage.

In other words, if a subject has used heroin within the last month, one may reasonably infer that they will use heroin again within the next month. Thus, we may use the recency of use as a reasonable forecasting signal, and more generally as indicative of potentially problematic usage patterns with respect to the drugs in question.

4.1 Generalization Experiment

Since our goal is to ultimately find a generalized model to predict concerning drug usage, we decided to conduct our own set of tests. From the original dataset, we reclassified the 7 categories of most recent self-reported drug usage into new categories. For each drug, we categorized each of the 7 original categories into No Concern, Slight Concern, or Very Concerning. The table shown in Figure 1 below describes this transformation.

	Substances					
Original Classification	Alcohol	legal highs	Nicotine	Cannabis	Caffeine	Chocolate
CL0	NC	NC	NC	NC	NC	NC
CL1	NC	NC	NC	NC	NC	NC
CL2	NC	NC	NC	NC	NC	NC
CL3	NC	NC	NC	NC	NC	NC
CL4	NC	NC	NC	NC	NC	NC
CL5	NC	NC	NC	NC	NC	NC
CL6	SC	SC	SC	SC	NC	NC
	Substances					
	Benzodiazepine	Cocaine	Crack	Ketamine	Amphetamines	Heroin
CL0	NC	NC	NC	NC	NC	NC
CL1	NC	NC	NC	NC	NC	NC
CL2	NC	NC	NC	NC	NC	NC
CL3	SC	SC	SC	SC	SC	SC
CL4	SC	SC	SC	SC	VC	VC
CL5	VC	VC	VC	VC	VC	VC
CL6	VC	VC	VC	VC	VC	VC
	Substances					
	Methadone	Magic shroom	VSA	Amyl Nitrite	Ecstasy	LSD
CL0	NC	NC	NC	NC	NC	NC
CL1	NC	NC	NC	NC	NC	NC
CL2	NC	NC	NC	NC	NC	NC
CL3	NC	NC	SC	SC	NC	NC
CL4	SC	SC	SC	SC	NC	SC
CL5	SC	SC	SC	SC	SC	SC
CL6	SC	SC	VC	SC	SC	SC

Figure 1: How each drug and its most recent self-reported use is categorized into the categories of "No Concern" (NC), "Slight Concern" (SC), and "Very Concerning" (VC)

We considered the following factors when determining which of the 3 categories a specific drug and a specific most recent usage should be put in:

- LD50 rating [6] - "the amount required to cause death (within a specified time frame) in 50% of nonhuman animals to which the drug is administered"
- Whether the use of a drug is likely to bring harm to the individual (for example: if a drug is highly addictive and/or highly correlated with deleterious health/lifestyle outcomes)

After the relabelling, the categories roughly align with societal norms regarding the harmfulness of each drug.

We then took the highest concern level that each individual is categorized with and produced a final label for each individual, such that if an individual's highest concern level is Slight Concern, then they would be labeled as Slight Concern. The table below in Figure 2 shows examples of this breakdown.

ID	...	Alcohol	Benzos	Caffeine	Cannabis	Choc	Coke	...	Final Label
1	...	0	0	0	0	0	0	...	0
2	...	0	0	0	0	0	1	...	1
3	...	1	0	0	0	0	0	...	1
...	...	...	...	...	...	...	...	...	...
23	...	0	0	0	0	0	0	...	0
24	...	1	2	0	0	0	0	...	2
25	...	0	0	0	0	0	0	...	1
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.

Figure 2: The categories of "No Concern", "Slight Concern", and "Very Concerning" are translated into categories "0", "1", and "2" for each drug. The "Final Label" column is the maximum concern value from all the drug columns for that specific sample

We then trained a random forest model that takes in all the existing demographic information (excluding country and ethnicity) and personality tests data as well as this final label column to produce a random forest model that predicts an individual's drug usage concern level.

The hyperparameters that we conducted a grid search for in developing our initial random forest model are nTree, mTry, and Cutoff. The mTry values iterated through are half the square root of the number of features, the square root of the number of features, and twice

the square root of the number of features. The following values were iterated through to find the optimal parameters:

nTree	**mTry**	**Cutoff**
1000, 2000, 5000	2, 4, 8	0.5,0.6,0.7

To find the most optimal parameters, we compared the F1 score of all the random forest models produced. We found that the below values produced the highest F1 score:

nTree	**mTry**	**Cutoff**
1000	4	0.6

The confusion matrix and F1 score of the best model with these hyperparameter is shown below in Figure 3.

	Predicted class 0	**Predicted class 1**	**Predicted class 2**
True class 0	258	243	26
True class 1	174	601	117
True class 2	23	293	150

Figure 3: The confusion matrix of the best random forest model using the optimal hyperparameter values

Recall (Class 0):

$$\frac{\text{Total Correctly Classified Class 0 cases}}{\text{Total Number of Actual Class 0 cases}} = \frac{258}{258+243+26} = \frac{258}{527} = 0.4896$$

Precision (Class 0):

$$\frac{\text{Total Correctly Classified Class 0 cases}}{\text{Total Number of Predicted Class 0 cases}} = \frac{258}{258+174+23} = \frac{258}{455} = 0.5670$$

F1 score (Class 0):

$$2 \times \frac{\text{Recall (Class 0)} \times \text{Precision (Class 0)}}{\text{Recall (Class 0)} + \text{Precision (Class 0)}} = 2 \times \frac{0.4896 \times 0.5670}{0.4896 + 0.5670} = 0.5254$$

$$\text{F1 score (Class 1)} = 0.5924$$

$$\text{F1 score (Class 2)} = 0.4953$$

$$\text{F1 score (average)} = \frac{0.5254 + 0.5924 + 0.4953}{3} = 0.5044$$

As shown by the F1 score and confusion matrix, the prediction model achieved lower than expected accuracy, particularly for class 2 ("Very Concerning").

The code and results from this experiment are available in the Appendix's Generalization Experiment at the end of this report. [9.1]

4.2 Classification Experiment

We also opted to try performing binary prediction of whether an individual is likely to be a drug user or not for specific drugs based on their demographic and personality tests results. For this, we followed the original authors' method at categorizing an individual as user/non-user. If an individual used a substance within the last decade, then they are categorized as a user. Individuals who reported all other usage frequencies are labeled as non-users.

We first preprocess the original dataset by categorizing the CL0 and CL1 labels to 0, meaning a non-user, and all the other CL2-CL6 labels as 1, meaning a user, for each drug per user. This converts the predicted usage of each drug into a binary classification formulation by aggregating through decade-based usage.

Figure 4 gives an example of the resulting dataset showing some drugs like Alcohol, Benzodiazepine, Caffeine, Cannabis, Chocolate and Cocaine:

ID	...	Alcohol	Benzos	Caffeine	Cannabis	Choc	Coke	...
1	...	0	0	0	0	0	0	...
2	...	0	0	0	0	0	1	...
3	...	1	0	0	0	0	0	...
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.

Figure 4: An example showing the dataset after data preprocessing is done

For this experiment, we continued using the already discovered best hyperparameters (based on the generalization experiment), shown here again:

nTree	mTry	Cutoff
1000	4	0.6

We performed this experiment on drugs that are of most interest and have the most balanced set of data. These drugs are: **Alcohol, Amphetamine, Benzodiazepine, Cannabis, Ecstasy, Legal Highs, Nicotine**. For this report's purpose, we will expand on the results for these drugs: **Cannabis** and **Ecstasy**.

4.2.1 Cannabis

For our Cannabis classification experiment, the random forest model developed using all personality and demographic features produced the confusion matrix shown in Figure 5.

The accuracy measures computed include overall accuracy, recall, precision, and F1 score for the classification experiment on Cannabis:

Overall Accuracy	Recall	Precision	F1 Score
80.01%	0.8838	0.8412	0.8620

	Predicted Positive	Predicted Negative
True Positive	1118	147
True Negative	211	409

Figure 5: Confusion matrix for cannabis user/non-user classification

For Cannabis classification experiment, we also produced the following plot in Figure 6 which shows the feature rankings of the features used in developing the model. Mean Decrease Accuracy (MDA) quantifies how the perturbation of individual feature values relates to the change in prediction error. Intuitively this shall inform us how important an individual feature is, to predict a given class. Mean Decrease Gini (Gini Index) measures the purity of a tree based on an individual feature. Purity relates to how well a certain feature splits the data well into the given set of prediction labels.

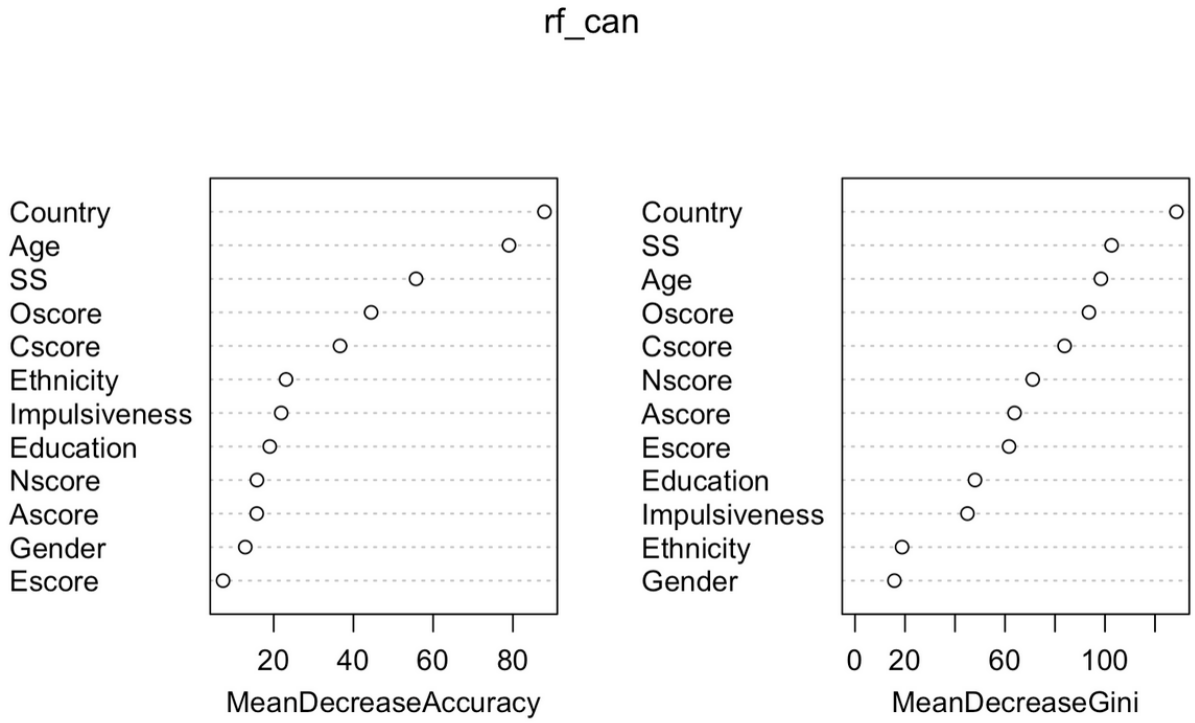


Figure 6: Feature ranking using importance metrics for Cannabis classification experiment

We also produced a table using the Random Forest Explainability (RFEX) model explainer for Cannabis as shown below in Figure 7. [7] The cumulative F1 score shows the gain in F1 score when we include features with lower ranks one-by-one. These cumulative F1 scores are calculated by using the features in seperate models in this order: (i) just use the top feature (ii) use the top two features (iii) use the top three features and so on.

The fourth and fifth columns show the average, standard deviation, minimum and maximum of all values for each feature per class. These statistical measures help us understand the distributions of the features in each class at a glance. Finally, the Cohen distance is a feature separation measure that incorporates the mean values and standard deviation values for understanding whether the distributions of a particular feature are likely to differentiate between the two classes with high confidence.

Feature Index	Feature Name	MDA Value	Cumulative F1 Score	AV/SD Positive Class; [Min, Max]	AV/SD Negative Class; [Min, Max]	Cohen Distance
1	Age	79.03	0.8146	-0.23/0.78 [-0.95,2.59]	0.58/0.81 [-0.95,2.59]	1.02
2	SS	55.70	0.8406	0.27/0.90 [-2.08,1.92]	-0.57/0.84 [-2.08,1.92]	0.97
3	Oscore	44.43	0.8337	0.24/0.95 [-2.86,2.90]	-0.50/0.90 [-3.27,2.15]	0.80
4	Cscore	36.63	0.8376	-0.19/0.98 [-3.46,3.00]	0.39/0.91 [-2.90,3.46]	0.61
5	Nscore	15.80	0.8481	0.1/1.01 [-3.46,3.27]	-0.20/0.93 [-2.76,3.27]	0.30

Figure 7: RFEX Model Explainer table for Cannabis

The feature ranking in the RFEX Model Explainer table is obtained by combining the MDA and Gini Index scores, using the average values of them and then ranking the features based on the highest average score values. We excluded Country, Ethnicity, and Gender in the RFEX ranking, even though we used them in this classification experiment as the measurement of Cohen distances doesn't make sense between categorical variables.

The feature ranks (starting from the highest ranked feature) are: (i) Age (ii) Sensation Seeking (iii) Oscore (iv) Cscore (v) Nscore. As can be seen in Figure 7, the cumulative F1 score stagnates at the top-2 features' cumulative F1 score. This means the model doesn't gain significant accuracy improvements by incorporating more features in the model. The first three features (Age, Sensation Seeking, and Oscore) have Cohen distances greater than 0.8, which is a good indicator that these features are likely to be good features for separation between the two classes. [8]

4.2.2 Ecstasy

For our Ecstasy classification experiment, the random forest model developed using all personality and demographic features produced the confusion matrix shown in Figure 8.

	Predicted Positive	Predicted Negative
True Positive	506	245
True Negative	252	882

Figure 8: confusion matrix for ecstasy user/non-user classification

The accuracy measures computed include overall accuracy, recall, precision, and F1 score:

Overall Accuracy	Recall	Precision	F1 Score
73.63%	0.6738	0.6675	0.6706

For Ecstasy classification experiment, we produced the plot showing the feature rankings of the features used in developing the model, which is shown in Figure 9. The top 5 ranked features (starting from the highest ranked feature) based on the average score values of the MDA and Gini Index are: (i) Age (ii) Sensation Seeking (iii) Oscore (iv) Cscore (v) Nscore.

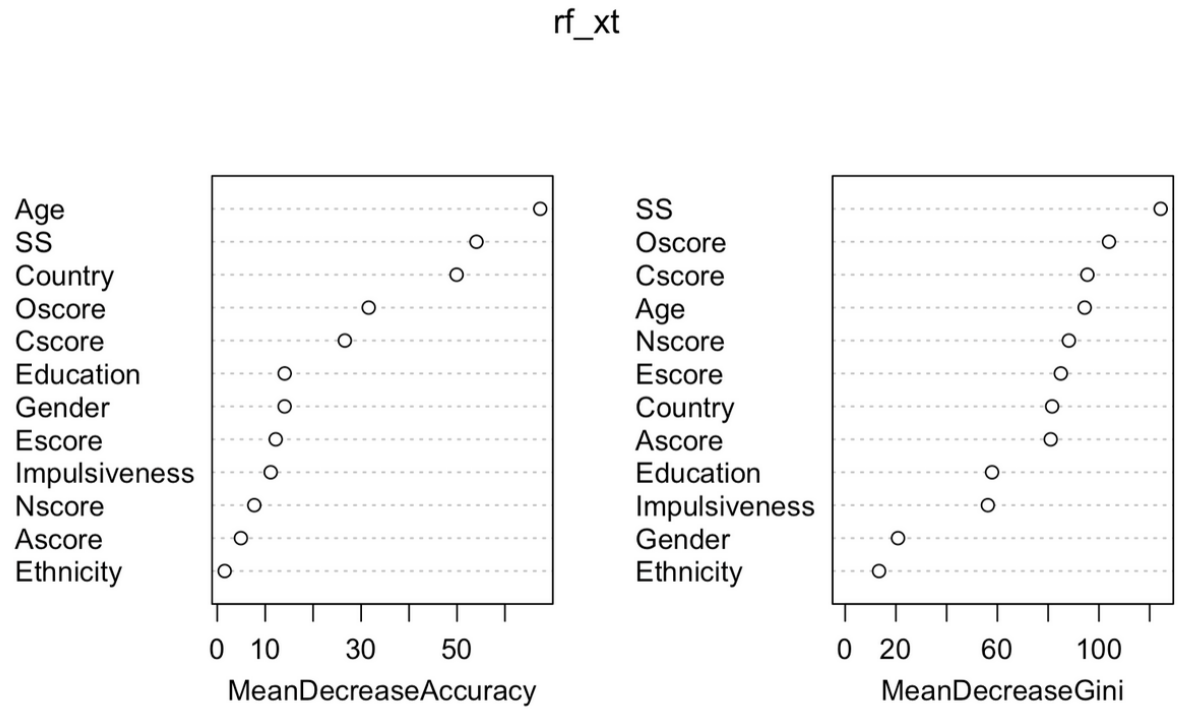


Figure 9: Feature ranking using importance metrics for Ecstasy classification experiment

We also produced the RFEX table for model explainability for Ecstasy as shown below in Figure 10.

Feature Index	Feature Name	MDA Value	Cumulative F1 Score	AV/SD Positive Class; [Min, Max]	AV/SD Negative Class; [Min, Max]	Cohen Distance
1	Age	67.35	0.57	-0.38/0.69 [-0.95,1.82]	0.31/0.88 [-0.95,2.59]	0.87
2	SS	54.05	0.64	0.47/0.84 [-2.08,1.92]	-0.32/0.91 [-2.08,1.92]	0.90
3	Oscore	31.58	0.66	0.36/0.94 [-2.86,2.90]	-0.24/0.96 [-3.27,2.90]	0.63
4	Cscore	26.60	0.66	-0.27/0.96 [-2.90,3.01]	0.18/0.98 [-3.46,3.46]	0.47
5	Nscore	7.73	0.65	0.12/1.00 [-2.76,3.27]	-0.08/0.99 [-3.46,3.27]	0.20

Figure 10: RFEX Model Explainer table for Ecstasy

As it can be seen in Figure 10, the cumulative F1 score stagnates at the top-2 features' cumulative F1 score. This means the model doesn't gain significant accuracy improvements by incorporating more features in the model, other than Age and Sensation Seeking. The first two features (Age and Sensation Seeking) have Cohen distances greater than 0.8, which, again, is a good indicator that these features are likely to be good features for separating the two classes.

As mentioned previously, we tested this classification experiment method using all the 12 features from the original dataset for a total of 7 drugs. The table below shows the accuracy measures for all 7 drugs data used in this experiment:

Drug	Overall Accuracy	Recall	Precision	F1 Score
Alcohol	92.07%	0.9967	0.9638	0.9800
Amphetamine	72.04%	0.5670	0.6230	0.5937
Benzos	69.87%	0.5722	0.6480	0.6077
Cannabis	81.01%	0.8838	0.8412	0.8620
Ecstasy	73.63%	0.6738	0.6675	0.6706
Legal Highs	78.94%	0.7178	0.7503	0.7337
Nicotine	72.94%	0.8655	0.7629	0.8110

The first part of the code and results from these classification experiments are available in Appendix 9.2 Decade-based Classification Experiment, Code and Results at the end of this report. [9.2]

The second part of the code and results from these classification experiments are available in Appendix 9.4 RFEX Code and Results, Code and Results at the end of this report. [9.4]

4.2.3 Classification with Subsampling

In addition to the tests above, we have tried undersampling to 'rebalance' datasets and similar user/non-user experiments were run with Random Forest models on the following drugs: **Cocaine, LSD, Mushrooms, Methadone, Crack, and Heroin**.

For these drugs, the balance of positive class (users) compared to negative class (non-users) resulted in less than 10% of the overall set being the positive class originally. Thus, undersampling the negative class in order to bring the positive class proportion back into balance was performed, in an attempt to achieve better predictive results.

The balance used was a 2:1 ratio of negative to positive class with appropriate undersampling for each drug tested.

Neither country of origin nor ethnicity were used as features in this experiment, though gender was retained as a feature in the mix for the model training.

The dataset was preprocessed for these drugs again as the user/non-user metric used here is based on drug usage within the last month, as this month-based frequency definition gives a reasonable balance of positive class before the undersampling procedure was performed.

The following table in Figure 11 shows the confusion matrix values which summarize the results.

Drug	Total Size of Set	Pred +, True +	Pred +, True -	Pred -, True +	Pred -, True -
Cocaine	**470**	77	82	60	251
LSD	**493**	96	70	59	268
Mushrooms	**470**	77	82	58	253
Methadone	**497**	77	94	48	278
Crack	**57**	10	10	5	32
Heroin	**145**	28	25	19	73

Figure 11: Confusion matrices for the 6 drugs

The table below in Figure 12 shows the accuracy measures for the same.

As can be seen, the accuracy measures with the subsampling experiment did not result in stellar F1 scores overall.

Drug	Overall Accuracy	Recall	Precision	F1 Score
Cocaine	69.79%	0.5620	0.4843	0.5203
LSD	73.83%	0.6194	0.5783	0.5981
Mushrooms	70.21%	0.5704	0.4843	0.5238
Methadone	71.43%	0.6160	0.4503	0.5203
Crack	73.68%	0.6667	0.5000	0.5714
Heroin	69.66%	0.5957	0.5283	0.5600

Figure 12: Accuracy measures for all 6 drugs after undersampling

4.3 Verification Experiment

Since the authors of the original paper provided the list of features per drug (Table 18 in [4]) that were used to obtain the most accurate model in their experiments, we decided to downsize the number of features used in developing the models based on their findings and see what the results would be.

In order to make a true attempt at replication, the decade based user/non-user criteria was used, and only the features that the original authors used to achieve their highest accuracy measures were used in the modeling experiments, taken from Table 18 from the original paper. It should be noted that we used Random Forest in our replication experiments, instead of Decision Trees which were used in the original paper.

Three different drugs were chosen to attempt replication, each with a differing number of features used (low, middle, high) corresponding to 2, 3, or 6 features used; two features were the minimum used in the original paper, and six was the maximum.

The following table in Figure 13 shows the confusion matrix values which summarize the results.

Drug	Total Size of Set	Pred +, True +	Pred +, True -	Pred -, True +	Pred -, True -
Crack	1885	6	185	72	1622
Ecstasy	1885	502	262	249	872
LSD	1885	304	253	178	1150

Figure 13: Confusion matrices for crack, ecstasy, and LSD using only the features used in the original paper

Drug	Number of Features Used	Overall Accuracy	Recall	Precision	F1 Score
Crack	**2**	86.37%	0.0769	0.0314	0.0446
Ecstasy	**3**	72.89%	0.6684	0.6571	0.6627
LSD	**6**	77.14%	0.6307	0.5458	0.5852

Figure 14: Accuracy measures for crack, ecstasy, and LSD using only the features used in the original paper

Drug	Original Sensitivity	Original Specificity	Attempt Sensitivity	Attempt Specificity
Crack	0.8063	0.7857	0.0769	0.8976
Ecstasy	0.7617	0.7716	0.6684	0.7690
LSD	0.8547	0.7756	0.6307	0.8197

Figure 15: Comparison of sensitivity and specificity values between the original paper and verification experiment

We note here that:

$$\text{Sensitivity} = \frac{\text{Number of True Positives}}{\text{Number of True Positives} + \text{Number of False Negatives}}$$

$$\text{Specificity} = \frac{\text{Number of True Negatives}}{\text{Number of True Negatives} + \text{Number of False Positives}}$$

As may be seen from the tables above, the attempt at replication resulted in much lower sensitivities than the original paper, though specificities were a bit higher in general. This could be due to the fact that if a model defaults to predicting negatives in general, the false positive rate is lower and thus specificity will appear higher.

The code and results from these verification experiments are available in the Appendix section for 'Feature-based Replication Attempt' at the end of this report. [9.5.2]

5 AI Ethics Audit and Results

AI systems are inherently complex, and given their far reaching impacts on society, especially when used in automated decision systems, a comprehensive audit process should be executed to ensure that AI principles such as fairness, safety, privacy, and explainability are upheld. While there are no standardized audit processes for AI systems, several papers have published general guidelines and best practices that should be followed. We modified the SMACTR approach proposed by Google's team to audit our project. [9] SMACTR stands for Scoping, Mapping, Artifact Collection, Testing, and Reflection. Inserted into this process are the discussion of the results from the experiments. We present our findings for each step of the auditing process in this section.

5.1 Scoping

The scoping phase defines the product, users, and scope of operations of our AI system. We also clarify the AI principles that we seek to uphold throughout our development and deployment process.

In the Introduction section, we provided the intended use case of this AI system: to predict at-risk drug use. Keeping in mind the AI principles of developing a safe system, we wish to minimize the number of false positives as well as pursue the goal of minimizing harm by allocating resources earlier to at-risk individuals or groups. The AI model developed is intended to be used in applications such that positive intervention efforts are deployed to reduce drug usage (or make earlier intervention possible), and it is specifically not intended to be used for exclusionary or discriminatory applications such as screening applicants for a job, financial aid, health care, or other opportunities.

Applications developed using our model should also not use the model's prediction as the sole determining factor in making decisions, nor should it be allowed to make automated decisions. The AI system should be used as a tool to help better inform human users so that they can make the final decision or use the results from this tool to augment their reasoning towards a final decision.

AI models should also be explainable, accountable, and transparent. By utilizing random forest as the machine learning algorithm, our team can explain predictions made from our AI model at a high level to human users through the use of feature rankings and explainability methods as described in section 4 of this report.

Lastly, we do not see any security or privacy issues resulting from our AI development. The dataset does not contain enough identifiable information to trace back to the original respondents. The information used for training can be considered non-personal, and no private information is exposed by the model in its application.

5.2 Mapping

The mapping stage identifies the roles and relations between all stakeholders that should be involved in the development and deployment process. We are concerned with identifying the correct users, subjects, developers, and designers etc., who would enable us to develop and audit this application in an ethical manner. Since the purpose of our model is to allow

resources to be allocated more efficiently to individuals who may be at higher risk of drug usage, we need to identify stakeholders who can help identify ground truth in our training data as well as those who are likely to utilize this tool.

Prior to beginning experimentation, it would be ideal to consult with drug addictions and rehabilitation counseling specialists or psychologists with experience working with individuals who engage in harmful drug usage. Their researches and experiences can help us in auditing the dataset and establishing ground truths in the data preprocessing step so that we can address any issues of bias or fairness and provide us the necessary expertise in determining whether a dataset is trustworthy or not. Consultation with these professionals can also help us understand the likely impact of a successful model and ensure that subject experts remain an integral part of our development process.

Another important group of stakeholders would be policy experts and community leaders. These groups are likely to be interested in such a tool so that positive intervention efforts can be initiated earlier and public health resources can be deployed more effectively. Their involvement can further shed light on how a tool like this would be utilized in the real world, and we can further educate them on the technology and the proper use cases that are suited for this model. Together, we can create a positive impact on society ranging from economic to health. A tool such as our AI model can help them make better and more impactful changes in our world.

5.3 Results

The code and results of our experiments are provided in the Appendix section of this report for transparency and reproducibility purposes. Individuals or groups looking to replicate, confirm, or expand upon our experiments can follow the steps we laid out throughout the report so that they can further investigate the link between personality measures and drug use.

Our generalization experiment shows that the model is not capable of accurately predicting slight or very concerning drug use, which is the original goal of our experiment. The average F1 score for all 3 classes is only 0.50. Combined with the fact that we know that the authors did not develop a generalized model but opted to create models for each drug, we therefore conclude that an accurate and trustworthy generalized machine learning model for drug usage concern level based on this dataset is unlikely to be found using random forest and this dataset alone.

The classification experiment attempted to look at the possibility of predicting usage of specific drugs. Even though we got somewhat acceptable F1 scores for some of the seven drugs tested, they might not be indicative of near-term usage because the predictions are based on decade-based definition of user/non-user. Even if this model was somehow useful, the model explainer contradicts our hypothesis of how the model might have internally put weights to the features. The results of RFEX show that the model can achieve optimal F1 scores based only on the top two features for the Cannabis and Ecstasy classification experiments. These features are Age and Sensation Seeking, and since most of the personality measures aren't informing the classification done by the model, it does not match the results of some of the previous research [1], which indicates that there are generalizable personality metrics which are associated with specific drug usage.

The classification experiments which were performed with the rebalanced datasets and the month-based user/non-user definitions also didn't yield accuracy measures up to the desired standard: the F1 scores ranged between 0.5 and 0.6. Of particular interest is that the predictions for the positive class didn't seem to be much better than random chance, judging from the class error rate and the resulting accuracy measures.

With respect to the attempt at replication/verification, the usage of Random Forest (as opposed to the paper's indicated Decision Trees) may play a role in the results. The accuracy measures overall were much worse than might be hoped for, with specificity results in a majority of cases (two thirds) markedly below those from the original paper.

5.4 Testing

After a model is developed, the auditing team enters the testing stage where the AI system is tested to ensure compliance with the stated ethical values listed in the scoping phase and evaluated according to the proper accuracy measures. Using random forest allows our team to assess the accuracy of developed machine learning models using accuracy measures such as precision, recall, and F1 score. These results are provided in the Experiment section and the Results subsection above. We also created a random forest model explainer to list feature ranking, cumulative F1 scores of the top few features, basic statistical measures of our class samples, and the Cohen Distance of the classes. These accuracy measures combine to give classification confidence measures of our resulting model.

Since our model did not produce satisfactory results for real-world deployment, we did not perform additional testing that would have exposed any vulnerabilities of our model. Such testing could be conducted by having stakeholders run through additional data that we were not included in our original dataset, or by testing synthetic data based on their own research and experiences. These would help enable us to develop risk analysis or mitigation plans on possible undesirable behavior, such as false positive rates.

Possible divergence from stated ethical principles could emerge if non-performant models were deployed in application and treated as if they were more accurate than present data would support. Thus it is important to emphasize that due to under-performance, these models, in their present state, are not recommended for product development or deployment.

5.5 Reflections

While we do not recommend these models to be deployed, we can reflect on possible future steps and how the case study goals can be better achieved through alternative approaches in the future. First, the demographics covered in the original dataset is not exhaustively/satisfactorily inclusive (especially when considering ethnicity and country of origin). This makes the model unsuitable for wider deployment even if the model works fine for the given demographics. Second, personality scores may be subjectively biased as outlined in the Data Audit section. Alternative ways of categorizing or gauging personality measures or obtaining such measures, perhaps in coordination with stakeholders such as psychologists or other external evaluators, could help produce a more grounded set of data.

It is somewhat concerning that for a number of the results, the 'Overall Accuracy' metric is not in line with actual performance of the model when looking at the other accuracy

measures, especially the more sensitive ones such as specificity, precision, or F1 score. The 'Overall Accuracy' measure would indicate to a non-domain expert that such a model is performing well, when it is actually the opposite case (in some cases, the F1 score and overall performance of the model could not be said to be much better than random chance).

So it is very important to communicate the overall picture of the accuracy measures, and that raw 'accuracy' is not the ultimate arbiter of a model's performance. It is also likely that if we had optimized the model to achieve high 'Overall Accuracy' instead of high F1 score, the model could have a higher tendency of producing even more biased results.

On a related note, if a model is trained to optimize for minimization of overall error, the easiest way for the model to minimize prediction error is to default to predictions for the most frequently occuring classes. This is potentially dangerous, as (for instance) a lung cancer detecting model which predicts 'no cancer' in 100% of trials would be accurate very often, but the consequences of missing a case are potentially catastrophic for the individual in question. This 'defaulting to the most numerous class' heuristic is especially pernicious when a set is imbalanced, i.e., when one class is much, much more frequently occuring than the other in a binary classification. The situation under examination across these experiments tends to be in this sort of a position, especially with respect to highly illegal and stigmatized drugs. By examining the dataset, one can see that the number of heroin or crack users in the dataset is miniscule compared to the number of non-users because that is representative of reality. In order to avoid this potential pitfall, the models we trained in the various experiments were optimized across hyperparameters for highest F1 scores.

Keeping in mind our goal to utilize the models for positive impact, we also need to balance and consider the impact of false positive and false negative classifications. Suppose that some of the models trained here were deployed, and they were used to try and determine if a government employee working in matters of national security was a potential risk for harmful drug usage. A false negative might not have an immediate negative impact on the individual, though it might be harmful to the organization's interests at large; while a false positive could potentially have a large, even career-killing impact on the employee in question, while this wouldn't have a considerable impact on the organization.

Thus accuracy is just one concern, while balancing the interests of the organization against the individual is another large concern. Accuracy is always going to be a desired quality, whereas which side the balance falls on as far as group vs. individual interests is highly context dependent. In general, we should consider that organizations are likely to default to deploying predictive models which maximize their own interests, even if the resulting dynamics disfavor individual members.

Our team also expected that if personality indeed matters for predicting at-risk drug usage, then personality measures should be ranked higher in the feature rankings for all models. It may be reasonable to also suspect that certain personality factors may play a larger role as predictive factors across certain or all drugs. Since our findings did not confirm our hypothesis, an additional consideration may be that besides personality, there are other factors that may influence drug use. Indeed, other researches have been done to find that psychological, financial, familial, social, and biological factors may influence likelihood of drug use. Obtaining a dataset that contains these additional factors could result in a more accurate machine learning model.

6 Summary and Recommendations

Our original intent was to develop a general machine learning model using random forest that would accurately predict an individual's risk level of using harmful drugs. The dataset used contained demographics information and personality measures of 1885 individuals as well as the individuals' most recent use of 18 different drugs. We then further preprocessed these data to focus our efforts on 3 main experiments using random forest.

Our generationalization experiment did not produce an accurate or reliable model, especially in the classes of interests "Slight Concern" and "Very Concerned". Creating different classification models for particular drugs of interest produced better predictions for specific drugs, but the level of confidence remains low as the most impactful features are not what we expected. Attempts to replicate results from the original paper also yielded lower than expected accuracy.

In the end, we are able to produce somewhat accurate predictions for only a few of the drugs, with the rest of the predictions no more accurate than a coin toss. None of the models are able to achieve high accuracy in areas of concern (high risk drugs vs. common substances of little to no concern like nicotine or chocolate), nor were any of the models able to replicate the results from the source paper.

Based on our results, we note that drug use prediction is not possible with this chosen dataset and model approach, and our models should not be deployed or trusted. It is unclear if drug use prediction is possible with high degree of confidence in general, as drug use is rare, especially with respect to 'hard' (highly illegal with little to no medical application and with high risk for abuse) drugs.

Though it is not the case, if a prospective model aimed at the problem under consideration was accurate, there would be more considerations around issues of ethics and oversight. For example, we recommend that a human being make the final decision. These ethical decisions should be made by domain experts in accordance with the AI Ethics guidelines developed during this research.

We do not recommend that our models should be used in any application as it is not possible at this time to assert that our models are reliable. Instead, we recommend pursuing future research using our learnings to focus on finding ways to make such predictions more accurate. For example, we can explore including more factors outside of personality measures, gathering data from a more diverse group, and/or utilizing other machine learning algorithms to perform similar tasks. These kinds of retrospective improvements based on our current findings may lead us towards a better model for achieving the goals as defined at the beginning of the report.

7 Contributions

7.1 Lawrence

- Team lead administrative tasks
  - setting meetings and goals
  - setting up Google Drive and collaborative space
- Generalization Experiment
- Rough draft of main content of report
- Reference research

7.2 Darshil

- Data Preprocessing for all experiments
- Classification Experiments
- RFEX calculations for model explainer
- A few replication experiments

7.3 Warren

- Rebalancing set experiments
- Paper analysis and group appraisals
- Paper replications efforts and analysis
- Presentation lead
- Slide deck structuring
- Report typesetting (LaTeX) and formatting

7.4 Everyone

- AI Audit Plan
- Assessment discussion and evaluations
- Report content writing, editing and formatting
- AI Ethics Analysis and Discussion

8 References

References

[1] Antonio Terracciano et al. "Five-Factor Model personality profiles of drug users". In: *BMC psychiatry* 8.1 (2008), pp. 1–10.

[2] Kate E Walton and Brent W Roberts. "On the relationship between substance use and personality traits: Abstainers are not maladjusted". In: *Journal of Research in Personality* 38.6 (2004), pp. 515–535.

[3] URL: https://archive.ics.uci.edu/ml/datasets/Drug+consumption+%5C%28quantified%5C%29.

[4] E. Fehrman et al. *The Five Factor Model of personality and evaluation of drug consumption risk*. 2015. DOI: 10.48550/ARXIV.1506.06297. URL: https://arxiv.org/abs/1506.06297.

[5] D. Petkovic. *AI and ML Overview [Lecture Slides]*. 2022.

[6] *APA Dictionary of Psychology*. 2022. URL: https://dictionary.apa.org/lethal-dose.

[7] Dragutin Petkovic et al. "Random Forest Model and Sample Explainer for Non-experts in Machine Learning–Two Case Studies". In: *International Conference on Pattern Recognition*. Springer. 2021, pp. 62–75.

[8] Gail M Sullivan and Richard Feinn. "Using effect size—or why the P value is not enough". In: *Journal of graduate medical education* 4.3 (2012), pp. 279–282.

[9] Inioluwa Deborah Raji et al. "Closing the AI accountability gap: Defining an end-to-end framework for internal algorithmic auditing". In: *Proceedings of the 2020 conference on fairness, accountability, and transparency*. 2020, pp. 33–44.

9 Appendix

9.1 Generalization Experiment

Generalization Experiment Code

```
# import necessary R packages
library(readr)
library(dplyr)
library(randomForest)
library(caret)

# read in sanitized drug data
drug_data <- read.csv("./Drug_Consumption_data_final_labels.csv")

glimpse(drug_data)

# categorize Label column as categorical
drug_data$Label <- as.factor(drug_data$Label)
drug_data$Gender <- as.factor(drug_data$Gender)

data_cols = drug_data[c('Age', 'Gender', 'Education', 'Nscore', 'Escore', 'Oscore', 'Ascore',
'Cscore','Impulsiveness','SS','Label')]

glimpse(data_cols)

# grid search parameters
# find optimal hyperparameter with regards to f1 score, not OOB

mtry <- c(2,4,8)
ntree <- c(1000,2000,5000)
cutoff <- c(0.5,0.6,0.7)

# Simple Testing just to ensure that the package works and understand how to interpret results
drug_model1.rf <- randomForest(Label ~.,
                      data = data_cols,
                      ntree = 2000,
                      mtry = 8,
                      CUTOFF = 0.6,
                      verbose = TRUE)

print(drug_model1.rf)

drug_model1.rf$confusion[1] # actual 0, predicted 0
drug_model1.rf$confusion[4] # actual 0, predicted 1
drug_model1.rf$confusion[7] # actual 0, predicted 2

drug_model1.rf$confusion[2] # actual 1, predicted 0
```

```
drug_model1.rf$confusion[5] # actual 1, predicted 1
drug_model1.rf$confusion[8] # actual 1, predicted 2

drug_model1.rf$confusion[3] # actual 2, predicted 0
drug_model1.rf$confusion[6] # actual 2, predicted 1
drug_model1.rf$confusion[9] # actual 2, predicted 2

# precision: total correctly classified positive cases/total number of predicted positive cases
# recall: total correctly classified positive Cases/total number of actual positive cases

# actual 0: 527
# actual 1: 892
# actual 2: 466

# rows is actual
# columns is predicted

p_0 =
drug_model1.rf$confusion[1]/(drug_model1.rf$confusion[1]+drug_model1.rf$confusion[2]+dru
g_model1.rf$confusion[3])
r_0 =
drug_model1.rf$confusion[1]/(drug_model1.rf$confusion[1]+drug_model1.rf$confusion[4]+dru
g_model1.rf$confusion[7])

p_1 =
drug_model1.rf$confusion[5]/(drug_model1.rf$confusion[4]+drug_model1.rf$confusion[5]+dru
g_model1.rf$confusion[6])
r_1 =
drug_model1.rf$confusion[5]/(drug_model1.rf$confusion[2]+drug_model1.rf$confusion[5]+dru
g_model1.rf$confusion[8])

p_2 =
drug_model1.rf$confusion[9]/(drug_model1.rf$confusion[7]+drug_model1.rf$confusion[8]+dru
g_model1.rf$confusion[9])
r_2 =
drug_model1.rf$confusion[9]/(drug_model1.rf$confusion[3]+drug_model1.rf$confusion[6]+dru
g_model1.rf$confusion[9])

p_0
r_0
p_1
r_1
p_2
r_2
```

```
f1_0 = (2*p_0*r_0)/(r_0+p_0)
f1_1 = (2*p_1*r_1)/(r_1+p_1)
f1_2 = (2*p_2*r_2)/(r_2+p_2)

f1_0
f1_1
f1_2

f1_avg = (f1_0+f1_1+f1_2)/3
f1_avg

mTryCol <- c()
nTreeCol <- c()
cutoffCol <- c()
f1_0Col <- c()
f1_1Col <- c()
f1_2Col <- c()
f1_avgCol <- c()

for(mTryVal in mtry)
{
  for(nTreeVal in ntree){
    for(cutoffVal in cutoff){
      print("-------------------------------")
      print(paste("mTry: ", mTryVal))
      print(paste("nTree: ", nTreeVal))
      print(paste("cutoff: ", cutoffVal))
      search_model <- randomForest(Label ~.,
                        data = data_cols,
                        ntree = nTreeVal,
                        mtry = mTryVal,
                        CUTOFF = cutoffVal,
                        verbose = TRUE,
                        importance = TRUE)

      print(search_model)

      p_0 =
search_model$confusion[1]/(search_model$confusion[1]+search_model$confusion[2]+search_
model$confusion[3])
      r_0 =
search_model$confusion[1]/(search_model$confusion[1]+search_model$confusion[4]+search_
model$confusion[7])
```

```
    p_1 =
search_model$confusion[5]/(search_model$confusion[4]+search_model$confusion[5]+search_
model$confusion[6])
    r_1 =
search_model$confusion[5]/(search_model$confusion[2]+search_model$confusion[5]+search_
model$confusion[8])

    p_2 =
search_model$confusion[9]/(search_model$confusion[7]+search_model$confusion[8]+search_
model$confusion[9])
    r_2 =
search_model$confusion[9]/(search_model$confusion[3]+search_model$confusion[6]+search_
model$confusion[9])

    f1_0 = (2*p_0*r_0)/(r_0+p_0)
    f1_1 = (2*p_1*r_1)/(r_1+p_1)
    f1_2 = (2*p_2*r_2)/(r_2+p_2)

    f1_avg = (f1_0+f1_1+f1_2)/3

    mTryCol <- c(mTryCol,mTryVal)
    nTreeCol <- c(nTreeCol,nTreeVal)
    cutoffCol <- c(cutoffCol,cutoffVal)
    f1_0Col <- c(f1_0Col,f1_0)
    f1_1Col <- c(f1_1Col,f1_1)
    f1_2Col <- c(f1_2Col,f1_2)
    f1_avgCol <- c(f1_avgCol,f1_avg)
    print("------------------------------")
   }
  }
}

# printing all F1 scores for all combinations of mTry, nTree, and cutoff for comparison and
verification purpose
df <- data.frame(mTryCol, nTreeCol, cutoffCol, f1_0Col, f1_1Col, f1_2Col, f1_avgCol)
print(df);

df[which.max(df$f1_avgCol),]
```

Generalization Experiment output

Run 1

```
30      2    1000      0.5 0.5179120 0.5927734 0.3812081 0.4972978
31      2    1000      0.6 0.5263158 0.6047642 0.3844086 0.5051629
32      2    1000      0.7 0.5114823 0.5970874 0.3829787 0.4971828
33      2    2000      0.5 0.5187500 0.6003899 0.3905013 0.5032137
34      2    2000      0.6 0.5330621 0.5993134 0.3770053 0.5031269
35      2    2000      0.7 0.5216495 0.5991232 0.3775100 0.4994276
36      2    5000      0.5 0.5262076 0.5999023 0.3760000 0.5007033
37      2    5000      0.6 0.5278351 0.5999028 0.3768506 0.5015295
38      2    5000      0.7 0.5288066 0.6010716 0.3785235 0.5028006
39      4    1000      0.5 0.5159301 0.5891089 0.3912484 0.4987625
40      4    1000      0.6 0.5066667 0.5835387 0.3864230 0.4922094
41      4    1000      0.7 0.5248227 0.5880020 0.4020888 0.5049712
42      4    2000      0.5 0.5252317 0.5901478 0.3875163 0.5009653
43      4    2000      0.6 0.5238095 0.5896806 0.3849155 0.4994685
44      4    2000      0.7 0.5300714 0.5912012 0.3942559 0.5051761
45      4    5000      0.5 0.5278351 0.5940887 0.3870130 0.5029789
46      4    5000      0.6 0.5138746 0.5897436 0.3953186 0.4996456
47      4    5000      0.7 0.5239551 0.5920079 0.3832021 0.4997217
48      8    1000      0.5 0.5191919 0.5824065 0.4114713 0.5043566
49      8    1000      0.6 0.5283401 0.5821268 0.3825417 0.4976695
50      8    1000      0.7 0.5260000 0.5862242 0.3943662 0.5021968
51      8    2000      0.5 0.5157360 0.5787106 0.3801020 0.4915162
52      8    2000      0.6 0.5271630 0.5840886 0.3974684 0.5029067
53      8    2000      0.7 0.5187437 0.5785786 0.3847134 0.4940119
54      8    5000      0.5 0.5231388 0.5804829 0.3908629 0.4981616
55      8    5000      0.6 0.5276939 0.5809045 0.3888183 0.4991389
56      8    5000      0.7 0.5232323 0.5813602 0.4000000 0.5015308
>
> df[which.max(df$f1_avgCol),]
   mTryCol nTreeCol cutoffCol   f1_0Col   f1_1Col   f1_2Col f1_avgCol
44      4    2000       0.7 0.5300714 0.5912012 0.3942559 0.5051761
```

Run 2

```
   mTryCol nTreeCol cutoffCol   f1_0Col   f1_1Col   f1_2Col f1_avgCol
1       2    1000       0.5 0.5196687 0.6027397 0.4105263 0.5109783
2       2    1000       0.6 0.5243523 0.5973646 0.3730159 0.4982443
3       2    1000       0.7 0.5227273 0.6048780 0.3989362 0.5088472
4       2    2000       0.5 0.5077720 0.5953307 0.3818425 0.4949817
5       2    2000       0.6 0.5181347 0.5937804 0.3748327 0.4955826
```

```
6       2    2000       0.7 0.5232678 0.6000974 0.3920000 0.5051218
7       2    5000       0.5 0.5274043 0.6028196 0.3887399 0.5063213
8       2    5000       0.6 0.5175983 0.6012627 0.3785235 0.4991282
9       2    5000       0.7 0.5231720 0.5977572 0.3716578 0.4975290
10      4    1000       0.5 0.5173116 0.5837892 0.4015444 0.5008817
11      4    1000       0.6 0.5292929 0.5917862 0.3873518 0.5028103
12      4    1000       0.7 0.5154639 0.5938735 0.4097938 0.5063771
13      4    2000       0.5 0.5234216 0.5925926 0.3853211 0.5004451
14      4    2000       0.6 0.5264228 0.5864811 0.3824289 0.4984443
15      4    2000       0.7 0.5295316 0.5919683 0.3968872 0.5061290
16      4    5000       0.5 0.5310275 0.5918570 0.4036223 0.5088356
17      4    5000       0.6 0.5245902 0.5965432 0.4005202 0.5072178
18      4    5000       0.7 0.5240532 0.5884672 0.3848168 0.4991124
19      8    1000       0.5 0.5227043 0.5786290 0.4000000 0.5004445
20      8    1000       0.6 0.5166498 0.5831663 0.3908046 0.4968736
21      8    1000       0.7 0.5217391 0.5799294 0.4060150 0.5025612
22      8    2000       0.5 0.5161943 0.5783133 0.3898734 0.4947937
23      8    2000       0.6 0.5243902 0.5822911 0.3862770 0.4976528
24      8    2000       0.7 0.5246231 0.5810675 0.4005070 0.5020659
25      8    5000       0.5 0.5252525 0.5824561 0.3923567 0.5000218
26      8    5000       0.6 0.5301205 0.5852674 0.3964646 0.5039508
27      8    5000       0.7 0.5288754 0.5839196 0.3934426 0.5020792
>
> df[which.max(df$f1_avgCol),]
  mTryCol nTreeCol cutoffCol   f1_0Col   f1_1Col   f1_2Col f1_avgCol
1       2    1000       0.5 0.5196687 0.6027397 0.4105263 0.5109783
```

Run 3

```
mTryCol nTreeCol cutoffCol   f1_0Col   f1_1Col   f1_2Col f1_avgCol
1       2    1000       0.5 0.5286160 0.6032978 0.3775100 0.5031413
2       2    1000       0.6 0.5292308 0.6033301 0.3877822 0.5067810
3       2    1000       0.7 0.5248963 0.6017358 0.3715847 0.4994056
4       2    2000       0.5 0.5207469 0.6018474 0.3898531 0.5041491
5       2    2000       0.6 0.5227273 0.5951220 0.3723404 0.4967299
6       2    2000       0.7 0.5309917 0.5999028 0.3758389 0.5022445
7       2    5000       0.5 0.5242518 0.5993201 0.3719677 0.4985132
8       2    5000       0.6 0.5182102 0.5970005 0.3665768 0.4939292
9       2    5000       0.7 0.5278351 0.5991232 0.3748327 0.5005970
10      4    1000       0.5 0.5309017 0.5836653 0.3896774 0.5014148
11      4    1000       0.6 0.5365854 0.5918164 0.4040921 0.5108313
12      4    1000       0.7 0.5199591 0.5875371 0.3839170 0.4971377
13      4    2000       0.5 0.5270684 0.5945679 0.3890339 0.5035568
14      4    2000       0.6 0.5265306 0.5898709 0.4072165 0.5078727
```

```
15      4    2000       0.7 0.5211123 0.5937962 0.3932292 0.5027125
16      4    5000       0.5 0.5210256 0.5880612 0.3828125 0.4972998
17      4    5000       0.6 0.5269583 0.5907966 0.3942559 0.5040036
18      4    5000       0.7 0.5200411 0.5890479 0.3896104 0.4995665
19      8    1000       0.5 0.5236657 0.5810000 0.3861004 0.4969220
20      8    1000       0.6 0.5253550 0.5856855 0.4100000 0.5070135
21      8    1000       0.7 0.5263158 0.5804196 0.3871795 0.4979716
22      8    2000       0.5 0.5236657 0.5863454 0.4000000 0.5033370
23      8    2000       0.6 0.5357503 0.5849246 0.3964422 0.5057057
24      8    2000       0.7 0.5176946 0.5814536 0.3944020 0.4978501
25      8    5000       0.5 0.5212982 0.5797392 0.3924051 0.4978142
26      8    5000       0.6 0.5252525 0.5817452 0.3918575 0.4996184
27      8    5000       0.7 0.5227964 0.5794486 0.3883249 0.4968566
>
> df[which.max(df$f1_avgCol),]
   mTryCol nTreeCol cutoffCol   f1_0Col   f1_1Col   f1_2Col f1_avgCol
11      4    1000       0.6 0.5365854 0.5918164 0.4040921 0.5108313
```

Run 4

```
mTryCol nTreeCol cutoffCol   f1_0Col   f1_1Col   f1_2Col f1_avgCol
1       2    1000       0.5 0.5175258 0.5976562 0.3776596 0.4976139
2       2    1000       0.6 0.5186722 0.6002900 0.3880597 0.5023406
3       2    1000       0.7 0.5221878 0.5930630 0.3793103 0.4981871
4       2    2000       0.5 0.5225410 0.5933202 0.3878628 0.5012413
5       2    2000       0.6 0.5211996 0.5987323 0.3829787 0.5009702
6       2    2000       0.7 0.5180599 0.5988372 0.3772049 0.4980340
7       2    5000       0.5 0.5288066 0.6001955 0.3829787 0.5039936
8       2    5000       0.6 0.5237113 0.6000000 0.3866667 0.5034593
9       2    5000       0.7 0.5212876 0.5996113 0.3818425 0.5009138
10      4    1000       0.5 0.5214724 0.5911330 0.3937008 0.5021021
11      4    1000       0.6 0.5261003 0.5927759 0.3937824 0.5042195
12      4    1000       0.7 0.5122449 0.5875931 0.3922581 0.4973653
13      4    2000       0.5 0.5194274 0.5883514 0.3864230 0.4980673
14      4    2000       0.6 0.5193483 0.5840357 0.3839170 0.4957670
15      4    2000       0.7 0.5241521 0.5929946 0.3896104 0.5022524
16      4    5000       0.5 0.5254582 0.5906838 0.3922078 0.5027833
17      4    5000       0.6 0.5230769 0.5890479 0.3932292 0.5017846
18      4    5000       0.7 0.5285714 0.5924827 0.3880208 0.5030250
19      8    1000       0.5 0.5282258 0.5855131 0.3873418 0.5003602
20      8    1000       0.6 0.5192698 0.5844612 0.4030418 0.5022576
21      8    1000       0.7 0.5232323 0.5748865 0.3864492 0.4948560
22      8    2000       0.5 0.5171026 0.5829146 0.3994911 0.4998361
23      8    2000       0.6 0.5178389 0.5835836 0.4045512 0.5019912
```

```
24      8    2000       0.7 0.5263158 0.5845070 0.4005038 0.5037755
25      8    5000       0.5 0.5257836 0.5858788 0.3928571 0.5015065
26      8    5000       0.6 0.5198777 0.5818363 0.3949045 0.4988728
27      8    5000       0.7 0.5237614 0.5861208 0.3856041 0.4984954
>
> df[which.max(df$f1_avgCol),]
   mTryCol nTreeCol cutoffCol   f1_0Col   f1_1Col   f1_2Col f1_avgCol
11      4    1000       0.6 0.5261003 0.5927759 0.3937824 0.5042195
```

Run 5

```
mTryCol nTreeCol cutoffCol   f1_0Col   f1_1Col   f1_2Col f1_avgCol
1       2    1000       0.5 0.5149948 0.5932039 0.3741588 0.4941192
2       2    1000       0.6 0.5139319 0.5955990 0.3862434 0.4985914
3       2    1000       0.7 0.5248963 0.6008708 0.3870095 0.5042589
4       2    2000       0.5 0.5202914 0.6017442 0.3704698 0.4975018
5       2    2000       0.6 0.5233645 0.6030978 0.3751687 0.5005437
6       2    2000       0.7 0.5155280 0.5953079 0.3905013 0.5004457
7       2    5000       0.5 0.5217391 0.6018474 0.3855422 0.5030429
8       2    5000       0.6 0.5165975 0.6004843 0.3697706 0.4956175
9       2    5000       0.7 0.5248963 0.6000974 0.3877822 0.5042586
10      4    1000       0.5 0.5244898 0.5879422 0.4112388 0.5078903
11      4    1000       0.6 0.5212121 0.5886454 0.3963731 0.5020769
12      4    1000       0.7 0.5325203 0.5952381 0.3948052 0.5075212
13      4    2000       0.5 0.5266393 0.5888174 0.3932730 0.5029099
14      4    2000       0.6 0.5239551 0.5854383 0.3870130 0.4988022
15      4    2000       0.7 0.5336049 0.5981216 0.3973856 0.5097040
16      4    5000       0.5 0.5262076 0.5907298 0.3901170 0.5023515
17      4    5000       0.6 0.5291113 0.5940594 0.3942931 0.5058213
18      4    5000       0.7 0.5238579 0.5877673 0.3953488 0.5023247
19      8    1000       0.5 0.5232323 0.5854637 0.3949045 0.5012001
20      8    1000       0.6 0.5237614 0.5809045 0.3969659 0.5005439
21      8    1000       0.7 0.5186680 0.5838323 0.3948387 0.4991130
22      8    2000       0.5 0.5282258 0.5860512 0.3872611 0.5005127
23      8    2000       0.6 0.5317221 0.5787620 0.3873418 0.4992753
24      8    2000       0.7 0.5256798 0.5800301 0.3826531 0.4961210
25      8    5000       0.5 0.5207700 0.5831663 0.3888183 0.4975849
26      8    5000       0.6 0.5187437 0.5797101 0.3836317 0.4940285
27      8    5000       0.7 0.5274949 0.5828343 0.3801020 0.4968104
>
> df[which.max(df$f1_avgCol),]
   mTryCol nTreeCol cutoffCol   f1_0Col   f1_1Col   f1_2Col f1_avgCol
15      4    2000       0.7 0.5336049 0.5981216 0.3973856  0.509704
```

[1] "mTry: 4"
[1] "nTree: 2000"
[1] "cutoff: 0.7"

Call:
 randomForest(formula = Label ~ ., data = data_cols, ntree = nTreeVal,   mtry = mTryVal, CUTOFF = cutoffVal, verbose = TRUE, importance = TRUE)
               Type of random forest: classification
                     Number of trees: 2000
No. of variables tried at each split: 4

        OOB estimate of error rate: 45.94%
Confusion matrix:
    0   1   2 class.error
0 262 236 29   0.5028463
1 169 605 118   0.3217489
2 24 290 152   0.6738197

Run 6

mTryCol	nTreeCol	cutoffCol	f1_0Col	f1_1Col	f1_2Col	f1_avgCol
1	2	1000	0.5	0.5258799	0.6001946	0.3850267	0.5037004
2	2	1000	0.6	0.5179120	0.5943304	0.3828648	0.4983691
3	2	1000	0.7	0.5312821	0.5953899	0.3835979	0.5034233
4	2	2000	0.5	0.5208333	0.5994180	0.3823529	0.5008681
5	2	2000	0.6	0.5119171	0.5977681	0.3924731	0.5007194
6	2	2000	0.7	0.5155280	0.6002914	0.3838926	0.4999040
7	2	5000	0.5	0.5308642	0.6027264	0.3736559	0.5024155
8	2	5000	0.6	0.5165975	0.6004843	0.3697706	0.4956175
9	2	5000	0.7	0.5217391	0.5979582	0.3775100	0.4990691
10	4	1000	0.5	0.5072464	0.5911330	0.4005168	0.4996321
11	4	1000	0.6	0.5254582	0.5924101	0.3952569	0.5043751
12	4	1000	0.7	0.5179856	0.5916049	0.3963731	0.5019879
13	4	2000	0.5	0.5220061	0.5910891	0.3906856	0.5012603
14	4	2000	0.6	0.5220966	0.5900345	0.3922078	0.5014463
15	4	2000	0.7	0.5159959	0.5878312	0.3853211	0.4963827
16	4	5000	0.5	0.5169928	0.5913129	0.3984476	0.5022511
17	4	5000	0.6	0.5184426	0.5911330	0.3874346	0.4990034
18	4	5000	0.7	0.5256674	0.5905045	0.3927649	0.5029789
19	8	1000	0.5	0.5238579	0.5851962	0.3860104	0.4983548
20	8	1000	0.6	0.5275827	0.5773300	0.3857868	0.4968998
21	8	1000	0.7	0.5185185	0.5783375	0.3969466	0.4979342
22	8	2000	0.5	0.5217391	0.5828743	0.3877551	0.4974562
23	8	2000	0.6	0.5392354	0.5814536	0.3815621	0.5007504

```
24      8    2000      0.7 0.5267407 0.5768653 0.3785166 0.4940409
25      8    5000      0.5 0.5238579 0.5855856 0.3939009 0.5011148
26      8    5000      0.6 0.5237614 0.5840000 0.3918054 0.4998556
27      8    5000      0.7 0.5292929 0.5800604 0.3954660 0.5016064
>
> df[which.max(df$f1_avgCol),]
  mTryCol nTreeCol cutoffCol   f1_0Col   f1_1Col   f1_2Col f1_avgCol
11      4    1000      0.6 0.5254582 0.5924101 0.3952569 0.5043751


[1] "mTry:  4"
[1] "nTree:  1000"
[1] "cutoff:  0.6"

Call:
 randomForest(formula = Label ~ ., data = data_cols, ntree = nTreeVal,      mtry = mTryVal,
CUTOFF = cutoffVal, verbose = TRUE, importance = TRUE)
               Type of random forest: classification
                     Number of trees: 1000
No. of variables tried at each split: 4

        OOB estimate of  error rate: 46.47%
Confusion matrix:
   0   1   2 class.error
0 258 243  26   0.5104364
1 174 601 117   0.3262332
2  23 293 150   0.6781116
```

9.2 Decade-based Classification Experiment, Code and Results

1. Importing the data and helper function(s)

```
library(readxl)
data <- read_excel("~/RStudio/CS_859_Team_Project/Dec1/Drug_Consumption_data_decade.xlsx")

f_cols = c("Gender", "Country", "Ethnicity", "Alcohol", "Amphet", "Benzos", "Cannabis", "Ecstasy", "Legalh", "Nicotine")

#drugs = list("Alcohol", "Amphet", "Benzos", "Cannabis", "Ecstasy", "Legalh", "Nicotine")

data[f_cols] <- lapply(data[f_cols], factor)

data
```

```
## # A tibble: 1,885 × 32
##       ID     Age Gender   Educa…¹ Country Ethni…² Nscore   Escore  Oscore Ascore
##    <dbl>   <dbl> <fct>      <dbl> <fct>   <fct>    <dbl>    <dbl>   <dbl>  <dbl>
##  1     1  0.498  0.48246  -0.0592 0.96082 0.126    0.313 -0.575   -0.583  -0.917
##  2     2 -0.0785 -0.48246  1.98   0.96082 -0.316… -0.678  1.94     1.44    0.761
##  3     3  0.498  -0.48246 -0.0592 0.96082 -0.316… -0.467  0.805   -0.847  -1.62
##  4     4 -0.952  0.48246   1.16   0.96082 -0.316… -0.149 -0.806   -0.0193  0.590
##  5     5  0.498  0.48246   1.98   0.96082 -0.316…  0.735 -1.63    -0.452  -0.302
##  6     6  2.59   0.48246  -1.23   0.24923 -0.316… -0.678 -0.300   -1.56    2.04
##  7     7  1.09   -0.48246  1.16   -0.570… -0.316… -0.467 -1.09    -0.452  -0.302
##  8     8  0.498  -0.48246 -1.74   0.96082 -0.316… -1.33   1.94    -0.847  -0.302
##  9     9  0.498  0.48246  -0.0592 0.24923 -0.316…  0.630  2.57    -0.976   0.761
## 10    10  1.82   -0.48246  1.16   0.96082 -0.316… -0.246  0.00332 -1.42    0.590
## # … with 1,875 more rows, 22 more variables: Cscore <dbl>, Impulsiveness <dbl>,
## #   SS <dbl>, Alcohol <fct>, Amphet <fct>, Amyl <dbl>, Benzos <fct>,
## #   Caff <dbl>, Cannabis <fct>, Choc <dbl>, Coke <dbl>, Crack <dbl>,
## #   Ecstasy <fct>, Heroin <dbl>, Ketamine <dbl>, Legalh <fct>, LSD <dbl>,
## #   Meth <dbl>, Mushrooms <dbl>, Nicotine <fct>, VSA <dbl>, Label <dbl>, and
## #   abbreviated variable names ¹Education, ²Ethnicity
```

```
alcohol = data[c("Age", "Gender", "Education", "Country", "Ethnicity", "Nscore", "Esc
ore", "Oscore", "Ascore", "Cscore", "Impulsiveness", "SS", "Alcohol")]

amphet = data[c("Age", "Gender", "Education", "Country", "Ethnicity", "Nscore", "Esco
re", "Oscore", "Ascore", "Cscore", "Impulsiveness", "SS", "Amphet")]

benzos = data[c("Age", "Gender", "Education", "Country", "Ethnicity", "Nscore", "Esco
re", "Oscore", "Ascore", "Cscore", "Impulsiveness", "SS", "Benzos")]

cannabis = data[c("Age", "Gender", "Education", "Country", "Ethnicity", "Nscore", "Es
core", "Oscore", "Ascore", "Cscore", "Impulsiveness", "SS", "Cannabis")]

ecstasy = data[c("Age", "Gender", "Education", "Country", "Ethnicity", "Nscore", "Esc
ore", "Oscore", "Ascore", "Cscore", "Impulsiveness", "SS", "Ecstasy")]

legalh = data[c("Age", "Gender", "Education", "Country", "Ethnicity", "Nscore", "Esco
re", "Oscore", "Ascore", "Cscore", "Impulsiveness", "SS", "Legalh")]

nicotine = data[c("Age", "Gender", "Education", "Country", "Ethnicity", "Nscore", "Es
core", "Oscore", "Ascore", "Cscore", "Impulsiveness", "SS", "Nicotine")]
```

```
f1 <- function(rf_model)

{
p = rf_model$confusion[4]/(rf_model$confusion[3]+rf_model$confusion[4])
r = rf_model$confusion[4]/(rf_model$confusion[2]+rf_model$confusion[4])

score = (2*p*r)/(r+p)

sprintf('Precision: %.4f, Recall: %.4f, F1 Score: %.4f', p, r,score)
}
```

2. Running a Random Forest

```
library(randomForest)
```

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
set.seed(365)

rf_oh <- randomForest(Alcohol~.,
                      data = alcohol,
                      importance = TRUE,
                      mtry = 4,
                      ntree = 1000,
                      CUTOFF = .6,
                      verbose = TRUE)
print(rf_oh)
```

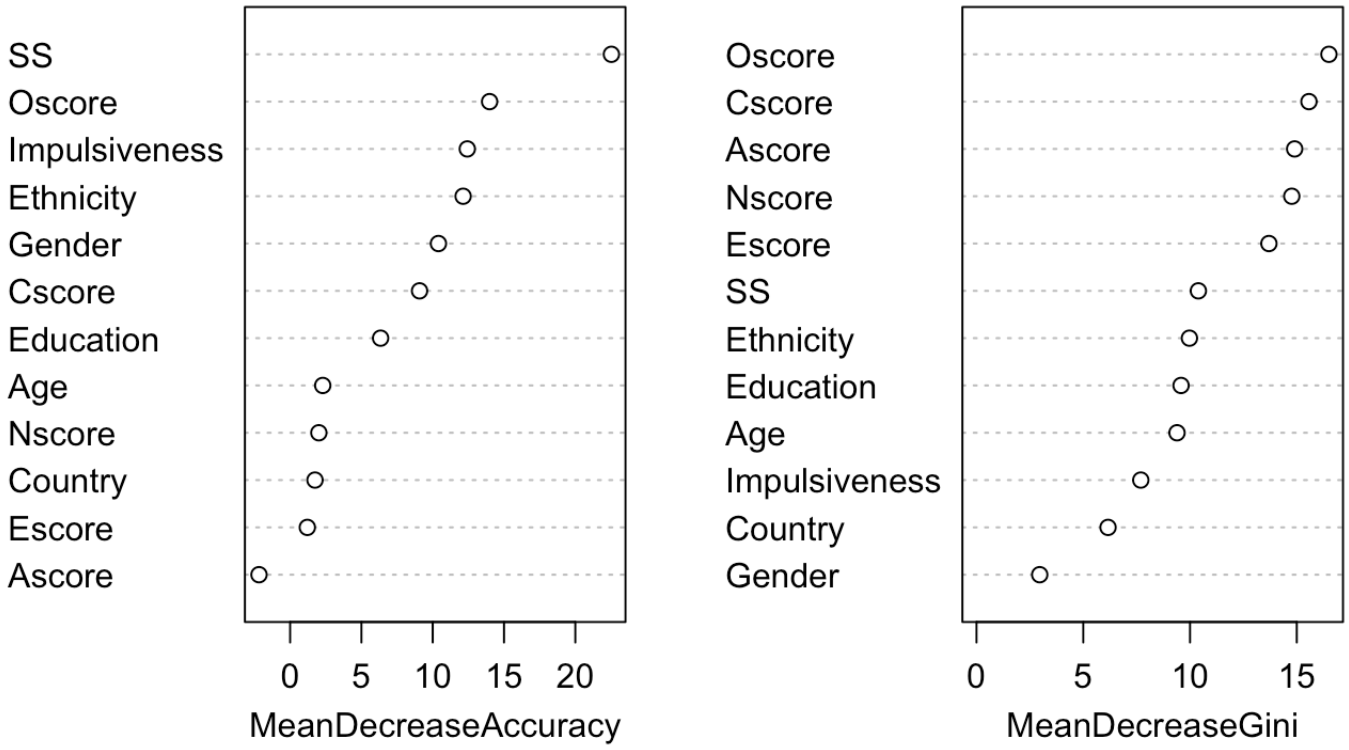
```
##
## Call:
##  randomForest(formula = Alcohol ~ ., data = alcohol, importance = TRUE,      mtry
= 4, ntree = 1000, CUTOFF = 0.6, verbose = TRUE)
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 3.93%
## Confusion matrix:
##   0    1 class.error
## 0 0   68 1.000000000
## 1 6 1811 0.003302146
```

```
f1(rf_oh) #Positive Class F1 Score function
```

```
## [1] "Precision: 0.9638, Recall: 0.9967, F1 Score: 0.9800"
```

```
varImpPlot(rf_oh)
```

rf_oh



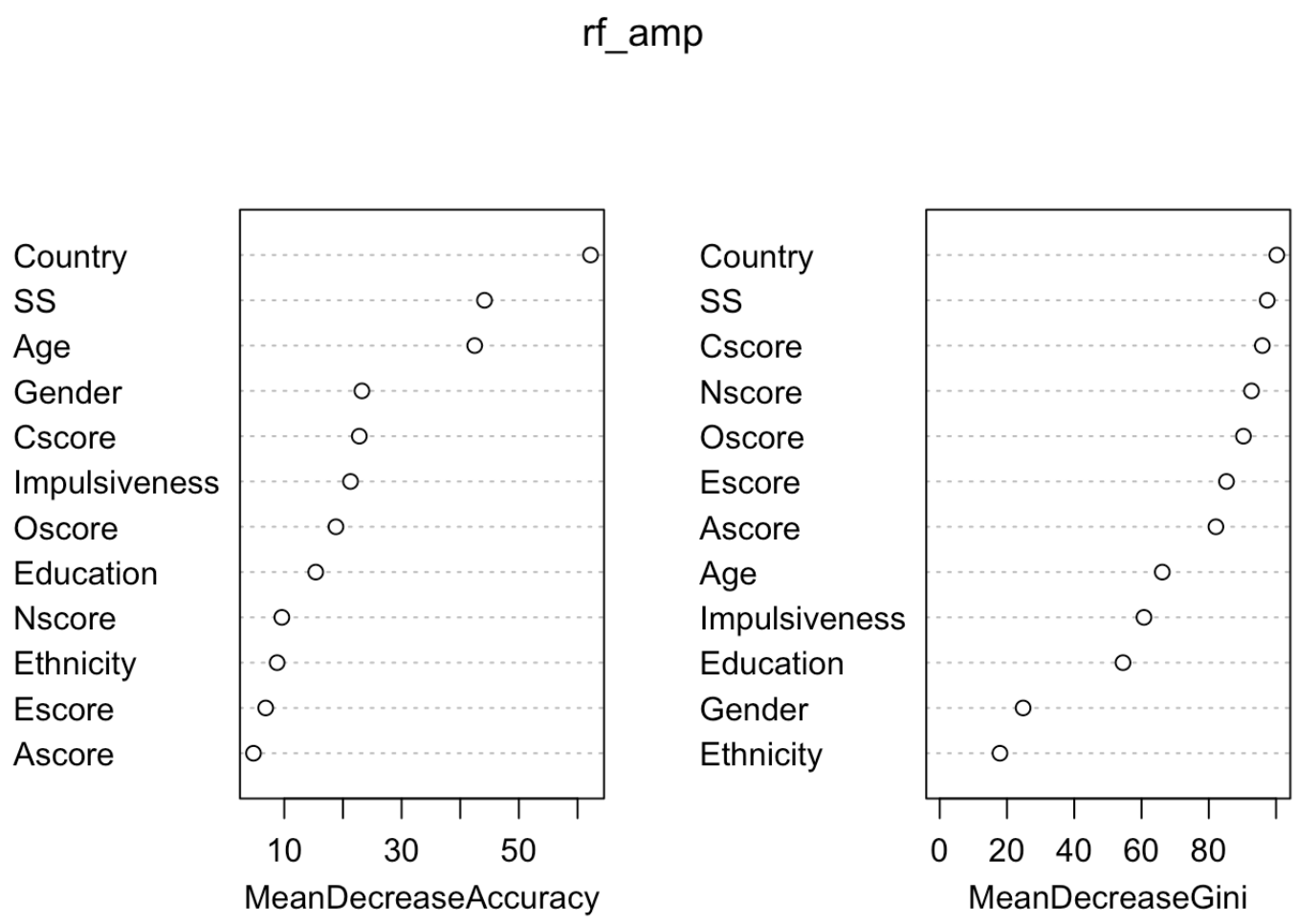

```
set.seed(365)

rf_amp <- randomForest(Amphet~.,
                       data = amphet,
                       importance = TRUE,
                       mtry = 4,
                       ntree = 1000,
                       CUTOFF = .6,
                       verbose = TRUE)
print(rf_amp)
```

```
##
## Call:
##  randomForest(formula = Amphet ~ ., data = amphet, importance = TRUE,      mtry =
4, ntree = 1000, CUTOFF = 0.6, verbose = TRUE)
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 27.96%
## Confusion matrix:
##     0   1 class.error
## 0 973 233   0.1932007
## 1 294 385   0.4329897
```

```
f1(rf_amp) #Positive Class F1 Score function
```

```
## [1] "Precision: 0.6230, Recall: 0.5670, F1 Score: 0.5937"
```

```
varImpPlot(rf_amp)
```



```
set.seed(365)

rf_ben <- randomForest(Benzos~.,
                     data = benzos,
                     importance = TRUE,
                     mtry = 4,
                     ntree = 1000,
                     CUTOFF = .6,
                     verbose = TRUE)
print(rf_ben)
```

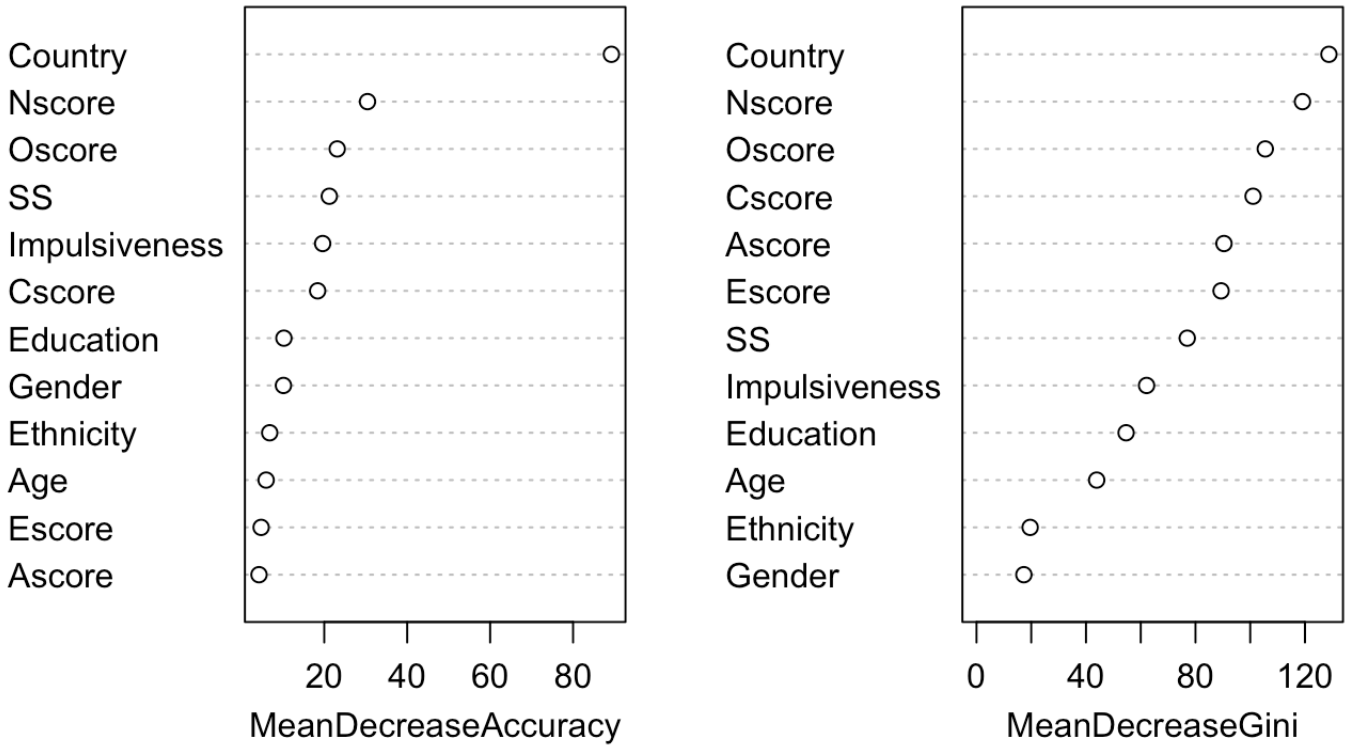
```
##
## Call:
##  randomForest(formula = Benzos ~ ., data = benzos, importance = TRUE,      mtry =
4, ntree = 1000, CUTOFF = 0.6, verbose = TRUE)
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 30.13%
## Confusion matrix:
##     0   1 class.error
## 0 877 239   0.2141577
## 1 329 440   0.4278283
```

```
f1(rf_ben) #Positive Class F1 Score function
```

```
## [1] "Precision: 0.6480, Recall: 0.5722, F1 Score: 0.6077"
```

```
varImpPlot(rf_ben)
```

rf_ben



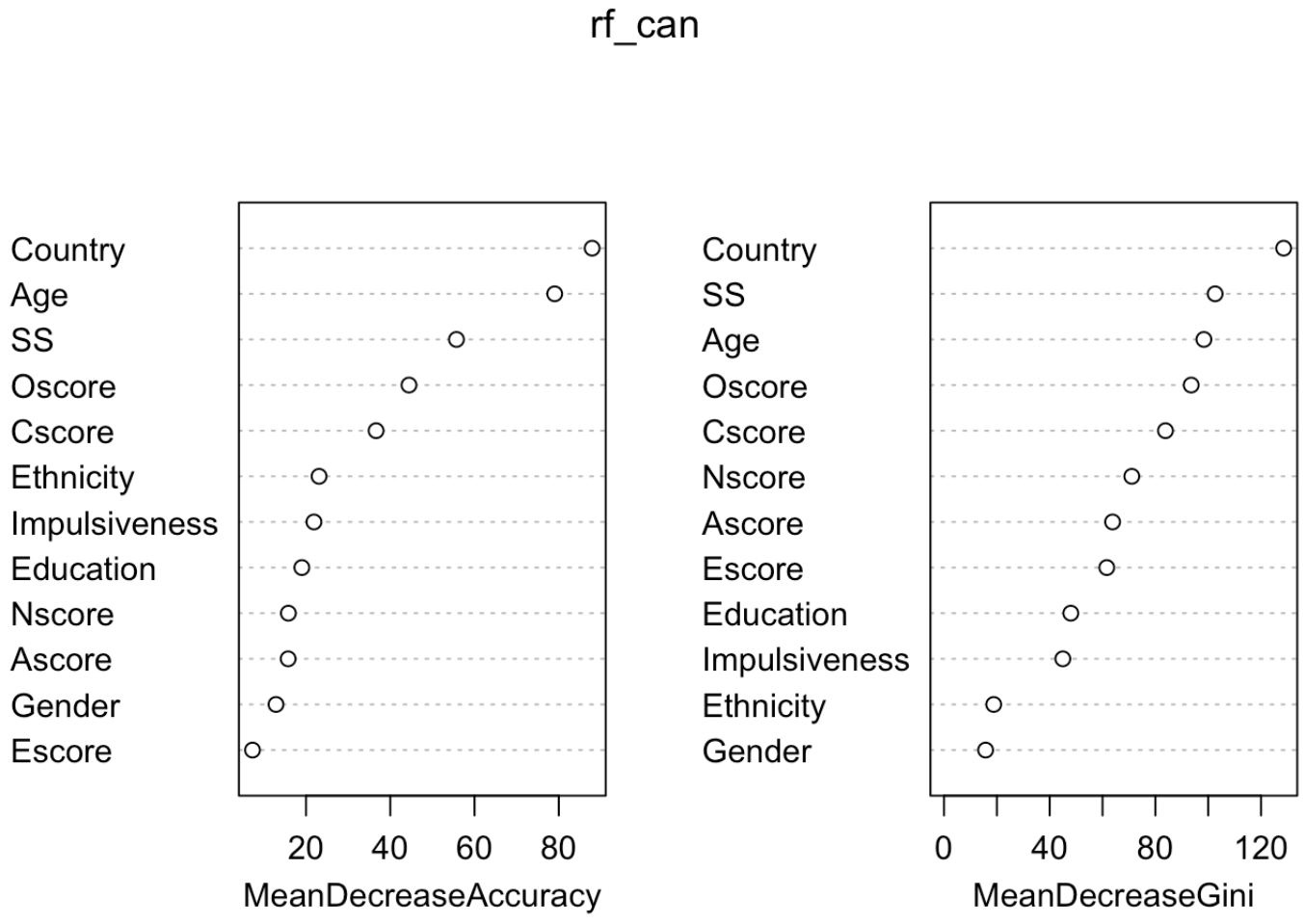

```
set.seed(365)

rf_can <- randomForest(Cannabis~.,
                       data = cannabis,
                       importance = TRUE,
                       mtry = 4,
                       ntree = 1000,
                       CUTOFF = .6,
                       verbose = TRUE)
print(rf_can)
```

```
##
## Call:
##  randomForest(formula = Cannabis ~ ., data = cannabis, importance = TRUE,      mtr
y = 4, ntree = 1000, CUTOFF = 0.6, verbose = TRUE)
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 18.99%
## Confusion matrix:
##     0    1 class.error
## 0 409  211   0.3403226
## 1 147 1118   0.1162055
```

```
f1(rf_can) #Positive Class F1 Score function
```

```
## [1] "Precision: 0.8412, Recall: 0.8838, F1 Score: 0.8620"
```

```
varImpPlot(rf_can)
```



```
importance(rf_can)
```

```
##                        0         1 MeanDecreaseAccuracy MeanDecreaseGini
## Age             84.89034 28.521380             79.03160         98.36428
## Gender          14.81144  3.069975             12.86435         15.73961
## Education       15.77715 10.264903             19.00554         47.98953
## Country        102.21447 25.740650             87.91879        128.55928
## Ethnicity       16.80846 17.502221             23.09239         18.81058
## Nscore          16.44409  5.083612             15.80379         71.07916
## Escore           7.27763  2.882402              7.29191         61.60701
## Oscore          37.26994 24.656971             44.43804         93.56095
## Ascore          15.33753  6.733953             15.74934         63.79891
## Cscore          34.57998 17.278796             36.63276         83.82265
## Impulsiveness   28.99711 -3.132806             21.87926         44.96434
## SS              56.90252 18.218207             55.70906        102.62726
```

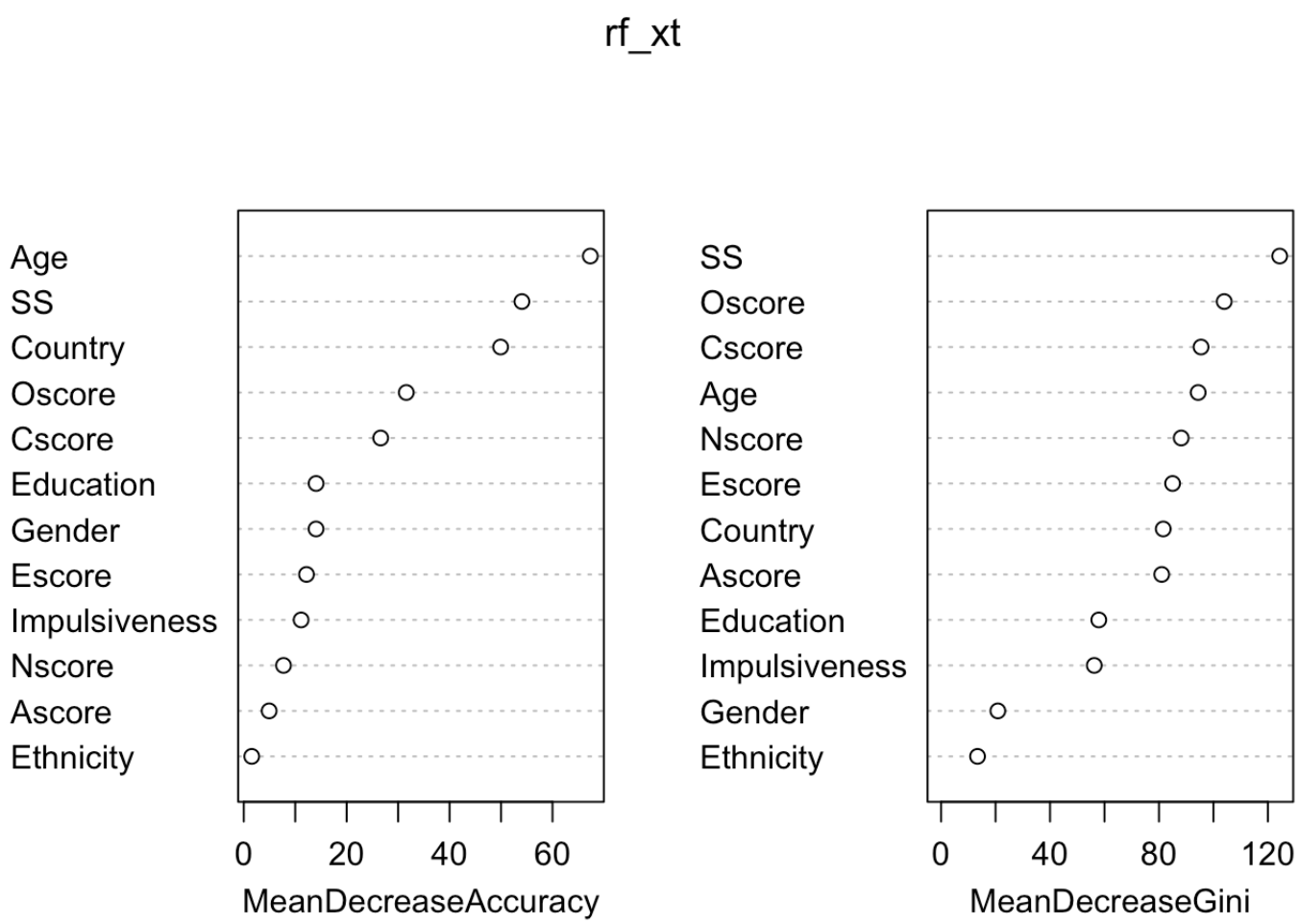
```
set.seed(365)

rf_xt <- randomForest(Ecstasy~.,
                      data = ecstasy,
                      importance = TRUE,
                      mtry = 4,
                      ntree = 1000,
                      CUTOFF = .6,
                      verbose = TRUE)
print(rf_xt)
```

```
##
## Call:
##  randomForest(formula = Ecstasy ~ ., data = ecstasy, importance = TRUE,      mtry
= 4, ntree = 1000, CUTOFF = 0.6, verbose = TRUE)
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 26.37%
## Confusion matrix:
##     0   1 class.error
## 0 882 252   0.2222222
## 1 245 506   0.3262317
```

```
f1(rf_xt) #Positive Class F1 Score function
```

```
## [1] "Precision: 0.6675, Recall: 0.6738, F1 Score: 0.6706"
```

```
varImpPlot(rf_xt)
```




```
importance(rf_xt)
```

```
##                      0          1 MeanDecreaseAccuracy MeanDecreaseGini
## Age          41.207234 52.3275797             67.347238         94.39676
## Gender       10.210008  9.7919470             14.047882         20.83609
## Education    15.627106  3.3282559             14.058821         57.90369
## Country      46.059033 14.9719800             49.916751         81.56859
## Ethnicity     2.982826 -0.5976505              1.550458         13.37620
## Nscore       12.039373 -2.0426361              7.725331         88.17218
## Escore       12.506637  3.7987430             12.190900         85.00611
## Oscore       27.269786 16.3574669             31.584402        103.97002
## Ascore        8.422231 -1.7688865              4.936217         81.01055
## Cscore       21.653827 14.2457882             26.608890         95.45419
## Impulsiveness  9.481943  5.1406684             11.147866         56.24333
## SS           41.029241 32.3477468             54.053326        124.33451
```

```
set.seed(365)

rf_lh <- randomForest(Legalh~.,
                      data = legalh,
                      importance = TRUE,
                      mtry = 4,
                      ntree = 1000,
                      CUTOFF = .6,
                      verbose = TRUE)
print(rf_lh)
```

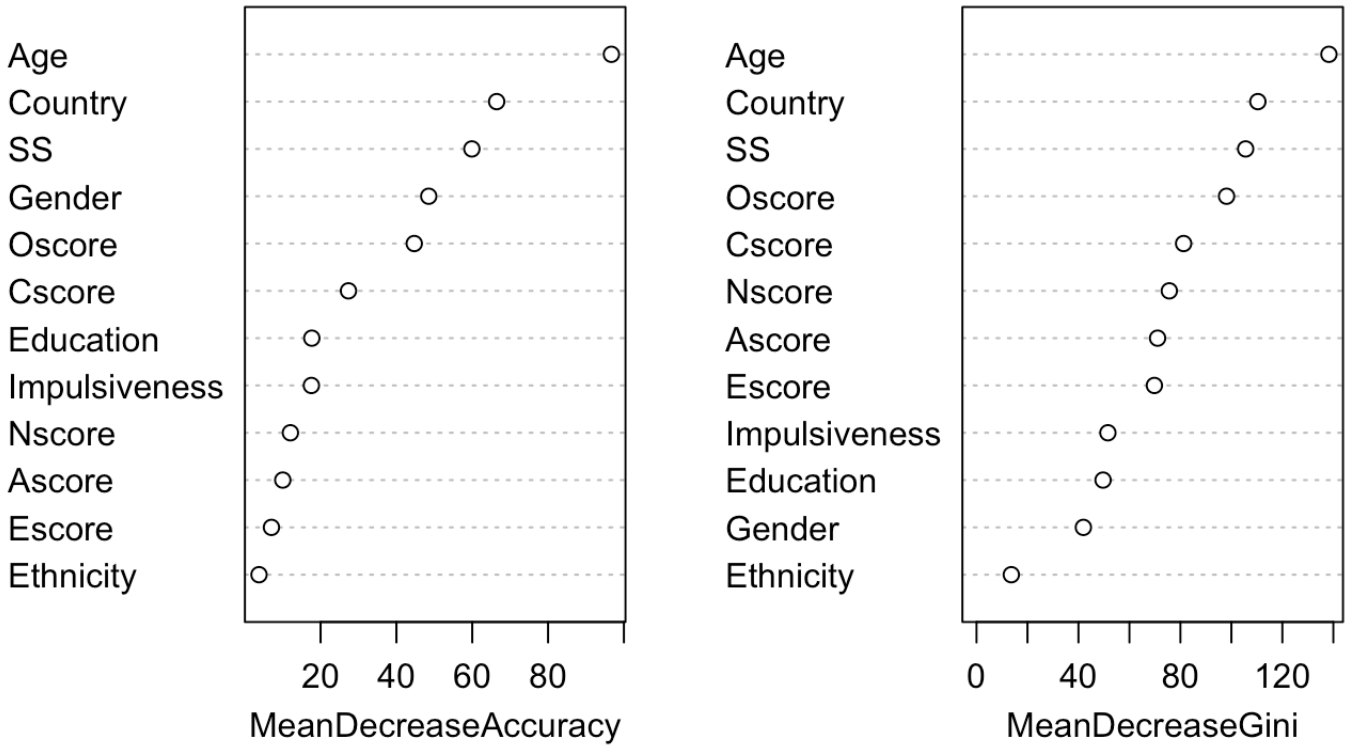
```
##
## Call:
##  randomForest(formula = Legalh ~ ., data = legalh, importance = TRUE,      mtry =
4, ntree = 1000, CUTOFF = 0.6, verbose = TRUE)
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 21.06%
## Confusion matrix:
##     0   1 class.error
## 0 941 182   0.1620659
## 1 215 547   0.2821522
```

```
f1(rf_lh) #Positive Class F1 Score function
```

```
## [1] "Precision: 0.7503, Recall: 0.7178, F1 Score: 0.7337"
```

```
varImpPlot(rf_lh)
```

rf_lh



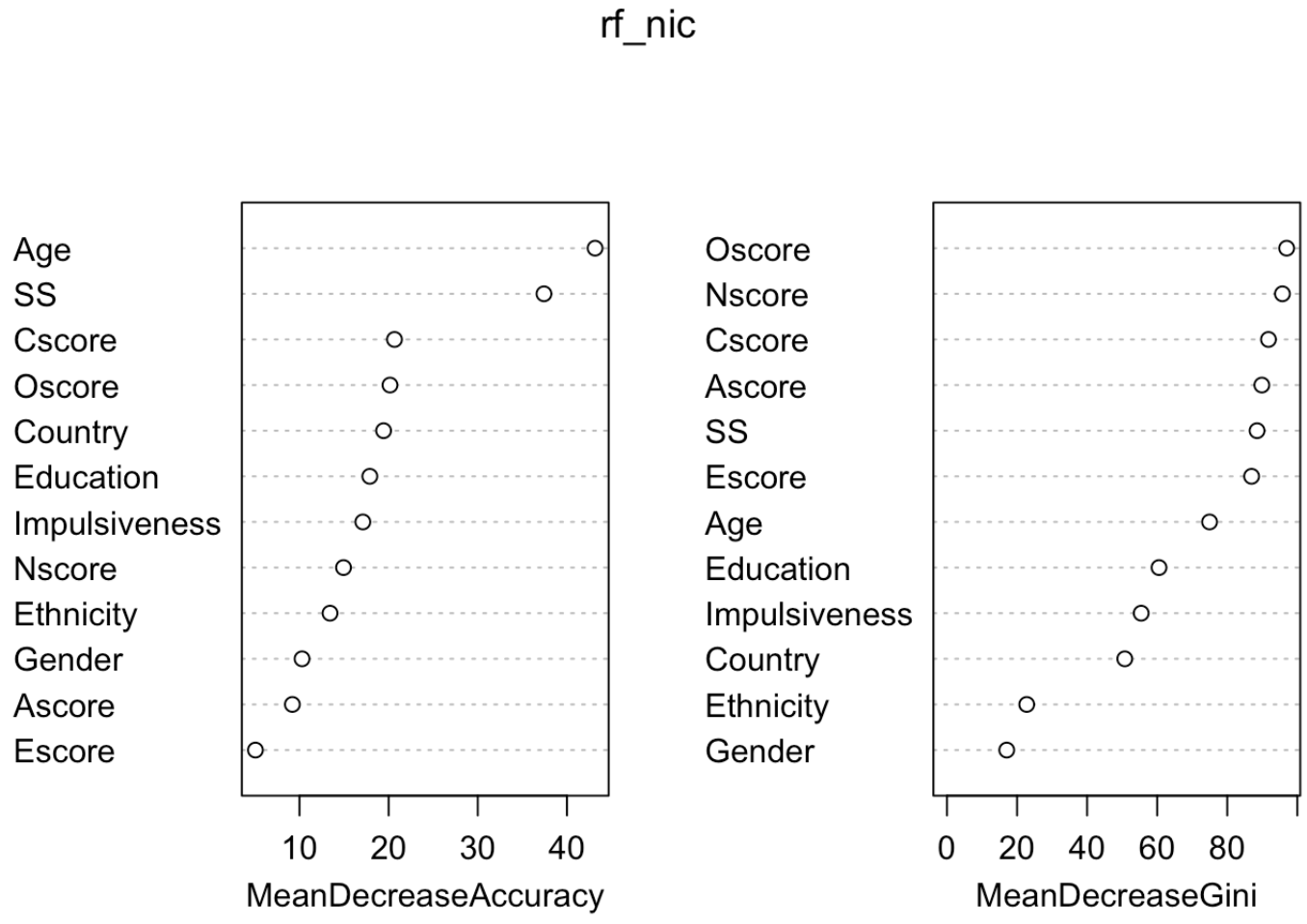

```
set.seed(365)

rf_nic <- randomForest(Nicotine~.,
                       data = nicotine,
                       importance = TRUE,
                       mtry = 4,
                       ntree = 1000,
                       CUTOFF = .6,
                       verbose = TRUE)
print(rf_nic)
```

```
##
## Call:
##  randomForest(formula = Nicotine ~ ., data = nicotine, importance = TRUE,      mtr
y = 4, ntree = 1000, CUTOFF = 0.6, verbose = TRUE)
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 27.06%
## Confusion matrix:
##     0    1 class.error
## 0 281  340   0.5475040
## 1 170 1094   0.1344937
```

```
f1(rf_nic) #Positive Class F1 Score function
```

```
## [1] "Precision: 0.7629, Recall: 0.8655, F1 Score: 0.8110"
```

```
varImpPlot(rf_nic)
```



9.3 Decade-based Replication Attempt, Code and Results

1. Importing the data and helper function(s)

```
library(readxl)
data <- read_excel("~/RStudio/CS_859_Team_Project/Dec1/Drug_Consumption_data_decade.xlsx")

f_cols = c("Gender", "Country", "Ethnicity", "Amphet", "Amyl", "Benzos", "Cannabis", "Crack", "Coke", "Ecstasy", "Legalh", "LSD", "Nicotine")

#drugs = list("Amphet", "Amyl", "Benzos", "Cannabis", "Coke", "Ecstasy", "Legalh", "LSD", "Nicotine")

data[f_cols] <- lapply(data[f_cols], factor)

data
```

```
## # A tibble: 1,885 × 32
##       ID     Age Gender   Educa…¹ Country Ethni…² Nscore   Escore  Oscore Ascore
##    <dbl>   <dbl> <fct>      <dbl> <fct>   <fct>    <dbl>    <dbl>   <dbl>  <dbl>
##  1     1  0.498  0.48246  -0.0592 0.96082 0.126    0.313 -0.575   -0.583  -0.917
##  2     2 -0.0785 -0.48246  1.98   0.96082 -0.316… -0.678  1.94     1.44    0.761
##  3     3  0.498  -0.48246 -0.0592 0.96082 -0.316… -0.467  0.805   -0.847  -1.62
##  4     4 -0.952  0.48246   1.16   0.96082 -0.316… -0.149 -0.806   -0.0193  0.590
##  5     5  0.498  0.48246   1.98   0.96082 -0.316…  0.735 -1.63    -0.452  -0.302
##  6     6  2.59   0.48246  -1.23   0.24923 -0.316… -0.678 -0.300   -1.56    2.04
##  7     7  1.09   -0.48246  1.16   -0.570… -0.316… -0.467 -1.09    -0.452  -0.302
##  8     8  0.498  -0.48246 -1.74   0.96082 -0.316… -1.33   1.94    -0.847  -0.302
##  9     9  0.498  0.48246  -0.0592 0.24923 -0.316…  0.630  2.57    -0.976   0.761
## 10    10  1.82   -0.48246  1.16   0.96082 -0.316… -0.246  0.00332 -1.42    0.590
## # … with 1,875 more rows, 22 more variables: Cscore <dbl>, Impulsiveness <dbl>,
## #   SS <dbl>, Alcohol <dbl>, Amphet <fct>, Amyl <fct>, Benzos <fct>,
## #   Caff <dbl>, Cannabis <fct>, Choc <dbl>, Coke <fct>, Crack <fct>,
## #   Ecstasy <fct>, Heroin <dbl>, Ketamine <dbl>, Legalh <fct>, LSD <fct>,
## #   Meth <dbl>, Mushrooms <dbl>, Nicotine <fct>, VSA <dbl>, Label <dbl>, and
## #   abbreviated variable names ¹Education, ²Ethnicity
```

```
amphet = data[c("Age", "Nscore", "Oscore", "Cscore", "Impulsiveness", "SS", "Amphet")
]

amyl = data[c("Nscore", "Oscore", "Cscore", "SS", "Amyl")]

benzos = data[c("Age", "Gender", "Nscore", "Escore", "Impulsiveness", "SS", "Benzos")
]

cannabis = data[c("Age", "Education", "Oscore", "Ascore", "Cscore", "Impulsiveness",
"Cannabis")]

crack = data[c("Escore", "Cscore", "Crack")]

coke = data[c("Age", "Oscore", "Ascore", "Impulsiveness", "SS", "Coke")]

ecstasy = data[c("Age", "Gender", "SS", "Ecstasy")]

legalh = data[c("Age", "Gender", "Oscore", "Ascore", "Cscore", "SS", "Legalh")]

lsd = data[c("Age", "Gender", "Nscore", "Escore", "Oscore", "Impulsiveness", "LSD")]

nicotine = data[c("Gender", "Nscore", "Escore", "Cscore", "Nicotine")]
```

```
f1 <- function(rf_model)

{
p = rf_model$confusion[4]/(rf_model$confusion[3]+rf_model$confusion[4])
r = rf_model$confusion[4]/(rf_model$confusion[2]+rf_model$confusion[4])
#sp = rf_model$confusion[4]/(rf_model$confusion[2]+rf_model$confusion[4])
#se = rf_model$confusion[1]/(rf_model$confusion[2]+rf_model$confusion[4])

score = (2*p*r)/(r+p)

#sprintf('Sensitivity: %.4f', se)
#sprintf('Specificity: %.4f', sp)
sprintf('Precision: %.4f, Recall: %.4f, F1 Score: %.4f',  p, r,score)
}
```

2. Running a Random Forest

```
library(randomForest)
```

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
set.seed(365)

rf_amp <- randomForest(Amphet~.,
                       data = amphet,
                       importance = TRUE,
                       mtry = 2,
                       ntree = 1000,
                       CUTOFF = .6,
                       verbose = TRUE)
print(rf_amp)
```

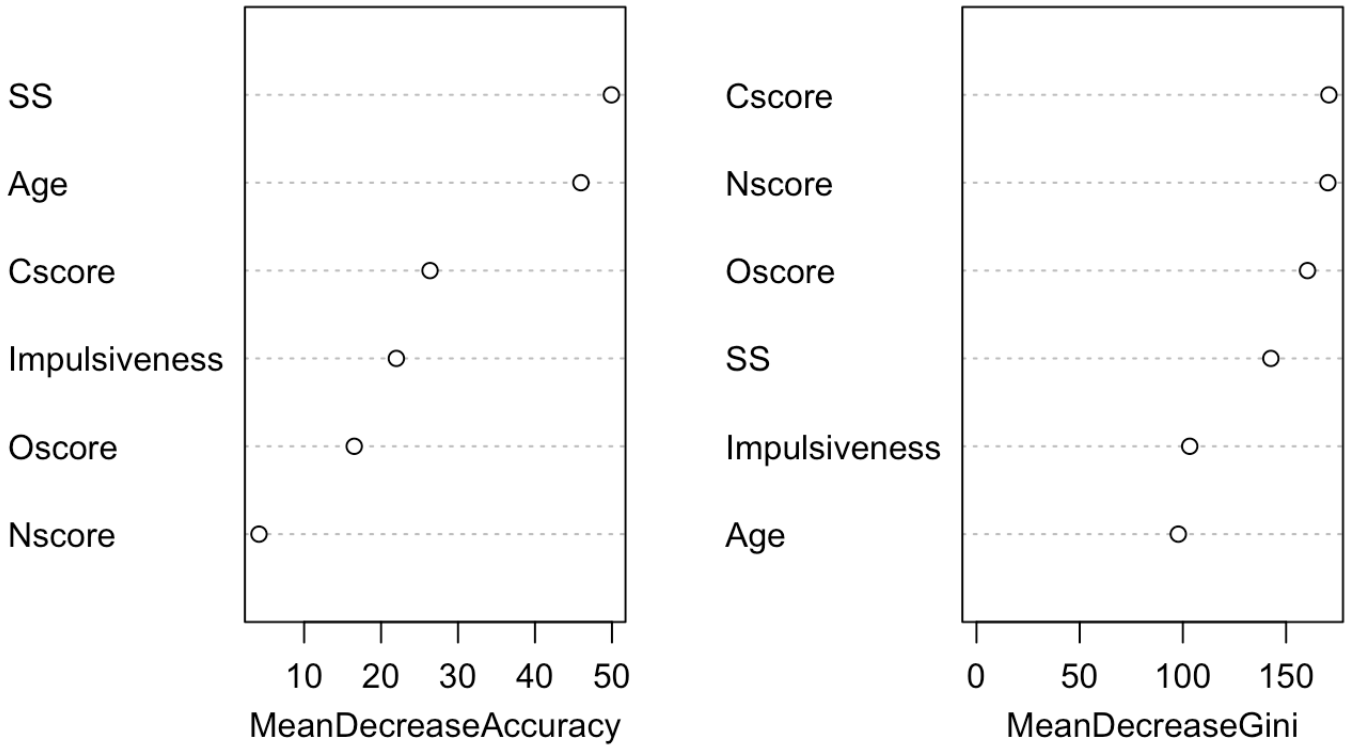
```
##
## Call:
##  randomForest(formula = Amphet ~ ., data = amphet, importance = TRUE,      mtry =
2, ntree = 1000, CUTOFF = 0.6, verbose = TRUE)
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 32.1%
## Confusion matrix:
##     0   1 class.error
## 0 942 264   0.2189055
## 1 341 338   0.5022091
```

```
f1(rf_amp) #Positive Class F1 Score function
```

```
## [1] "Precision: 0.5615, Recall: 0.4978, F1 Score: 0.5277"
```

```
varImpPlot(rf_amp)
```

rf_amp



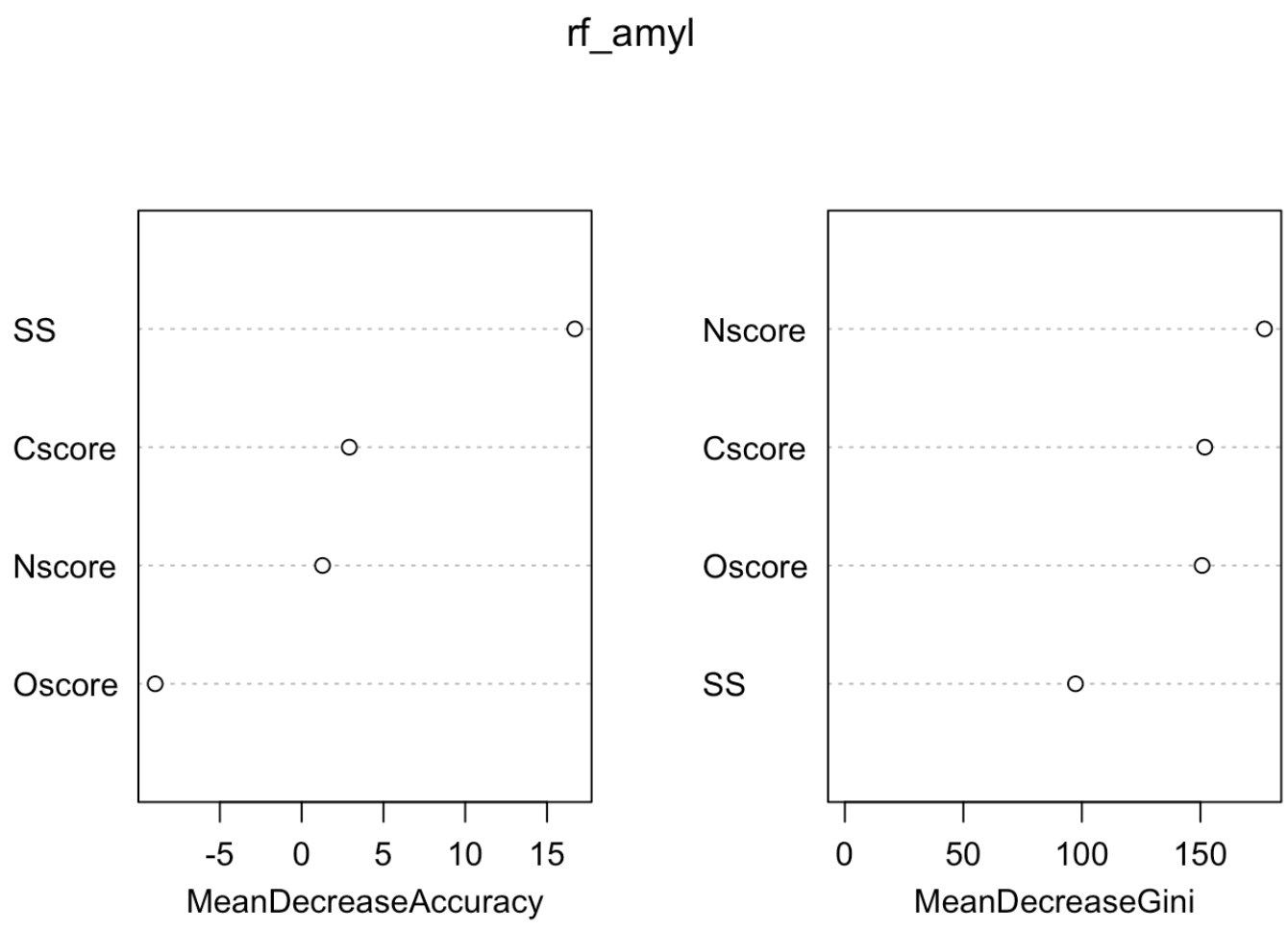

```
set.seed(365)

rf_amyl <- randomForest(Amyl~.,
                     data = amyl,
                     importance = TRUE,
                     mtry = 2,
                     ntree = 1000,
                     CUTOFF = .6,
                     verbose = TRUE)
print(rf_amyl)
```

```
##
## Call:
##  randomForest(formula = Amyl ~ ., data = amyl, importance = TRUE,      mtry = 2, n
tree = 1000, CUTOFF = 0.6, verbose = TRUE)
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 22.55%
## Confusion matrix:
##      0  1 class.error
## 0 1436 79  0.05214521
## 1  346 24  0.93513514
```

```
f1(rf_amyl) #Positive Class F1 Score function
```

```
## [1] "Precision: 0.2330, Recall: 0.0649, F1 Score: 0.1015"
```

```
varImpPlot(rf_amyl)
```



```
set.seed(365)

rf_ben <- randomForest(Benzos~.,
                     data = benzos,
                     importance = TRUE,
                     mtry = 2,
                     ntree = 1000,
                     CUTOFF = .6,
                     verbose = TRUE)
print(rf_ben)
```

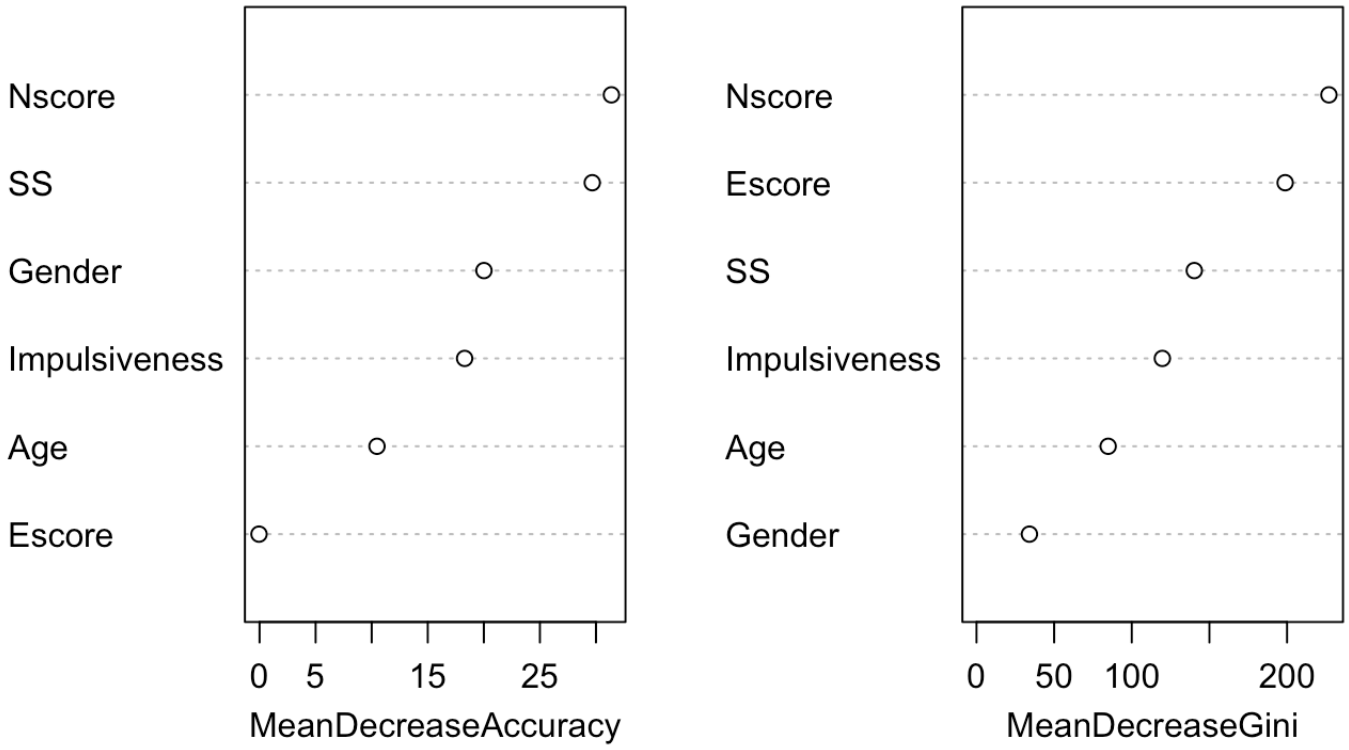
```
##
## Call:
##  randomForest(formula = Benzos ~ ., data = benzos, importance = TRUE,      mtry =
2, ntree = 1000, CUTOFF = 0.6, verbose = TRUE)
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 35.49%
## Confusion matrix:
##     0   1 class.error
## 0 849 267   0.2392473
## 1 402 367   0.5227568
```

```
f1(rf_ben) #Positive Class F1 Score function
```

```
## [1] "Precision: 0.5789, Recall: 0.4772, F1 Score: 0.5232"
```

```
varImpPlot(rf_ben)
```

rf_ben



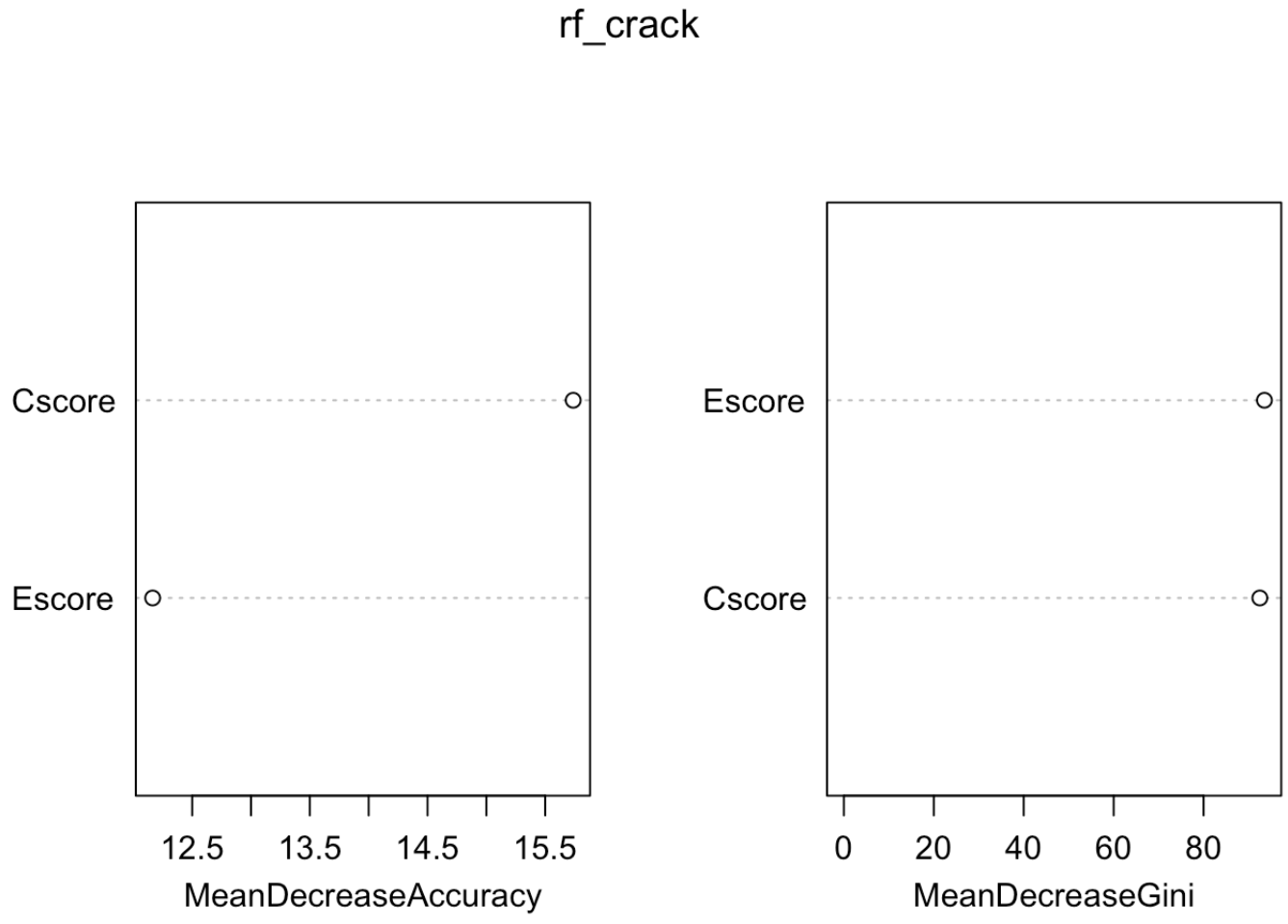

```
set.seed(365)

rf_crack <- randomForest(Crack~.,
                     data = crack,
                     importance = TRUE,
                     mtry = 2,
                     ntree = 1000,
                     CUTOFF = .6,
                     verbose = TRUE)
print(rf_crack)
```

```
##
## Call:
##  randomForest(formula = Crack ~ ., data = crack, importance = TRUE,      mtry = 2,
ntree = 1000, CUTOFF = 0.6, verbose = TRUE)
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 12.63%
## Confusion matrix:
##      0  1 class.error
## 0 1644 50  0.02951594
## 1  188  3  0.98429319
```

```
f1(rf_crack) #Positive Class F1 Score function
```

```
## [1] "Precision: 0.0566, Recall: 0.0157, F1 Score: 0.0246"
```

```
varImpPlot(rf_crack)
```



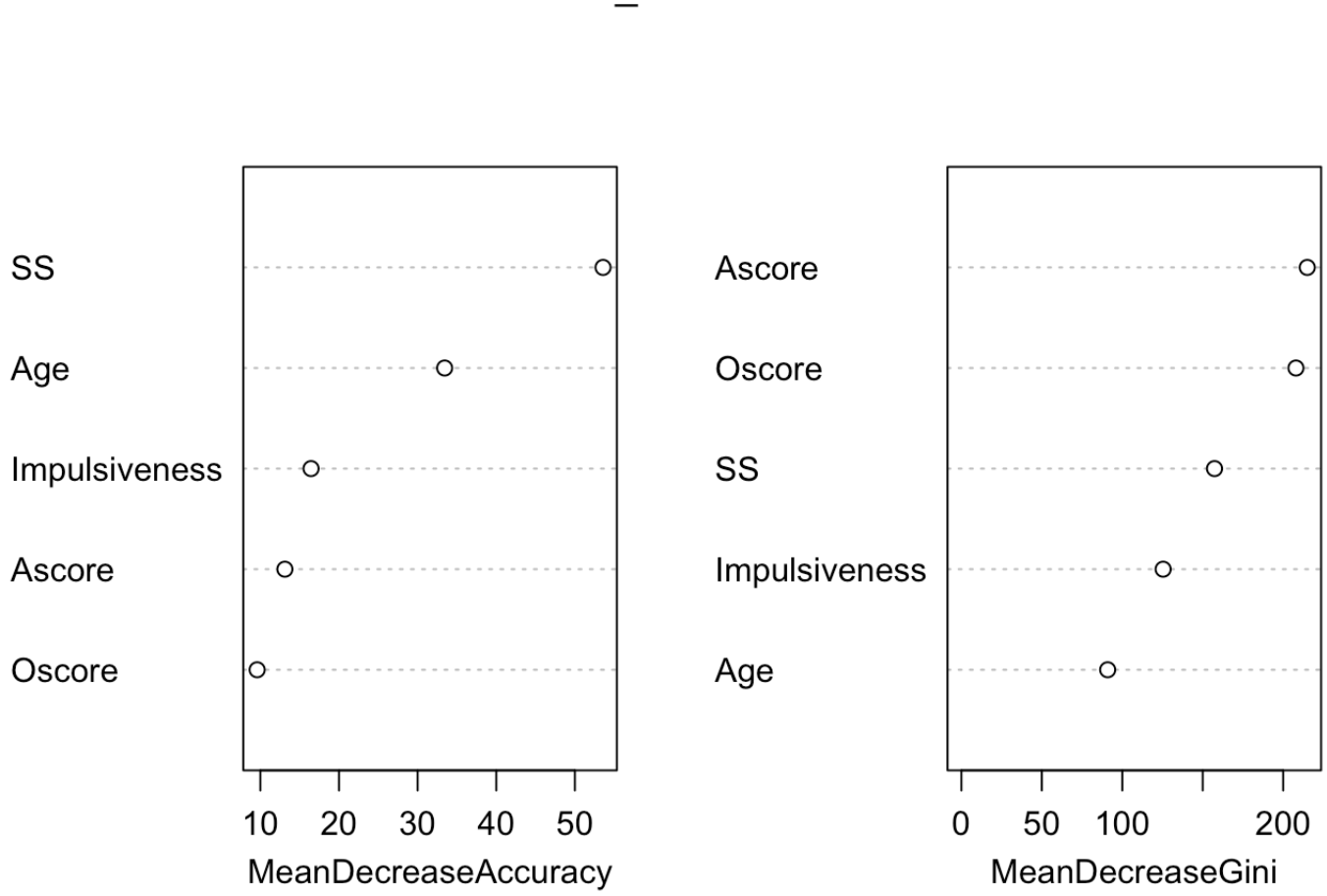
```
set.seed(365)

rf_coke <- randomForest(Coke~.,
                    data = coke,
                    importance = TRUE,
                    mtry = 2,
                    ntree = 1000,
                    CUTOFF = .6,
                    verbose = TRUE)
print(rf_coke)
```

```
##
## Call:
##  randomForest(formula = Coke ~ ., data = coke, importance = TRUE,      mtry = 2, n
tree = 1000, CUTOFF = 0.6, verbose = TRUE)
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 35.97%
## Confusion matrix:
##     0   1 class.error
## 0 924 274   0.2287145
## 1 404 283   0.5880640
```

```
f1(rf_coke) #Positive Class F1 Score function
```

```
## [1] "Precision: 0.5081, Recall: 0.4119, F1 Score: 0.4550"
```

```
varImpPlot(rf_coke)
```



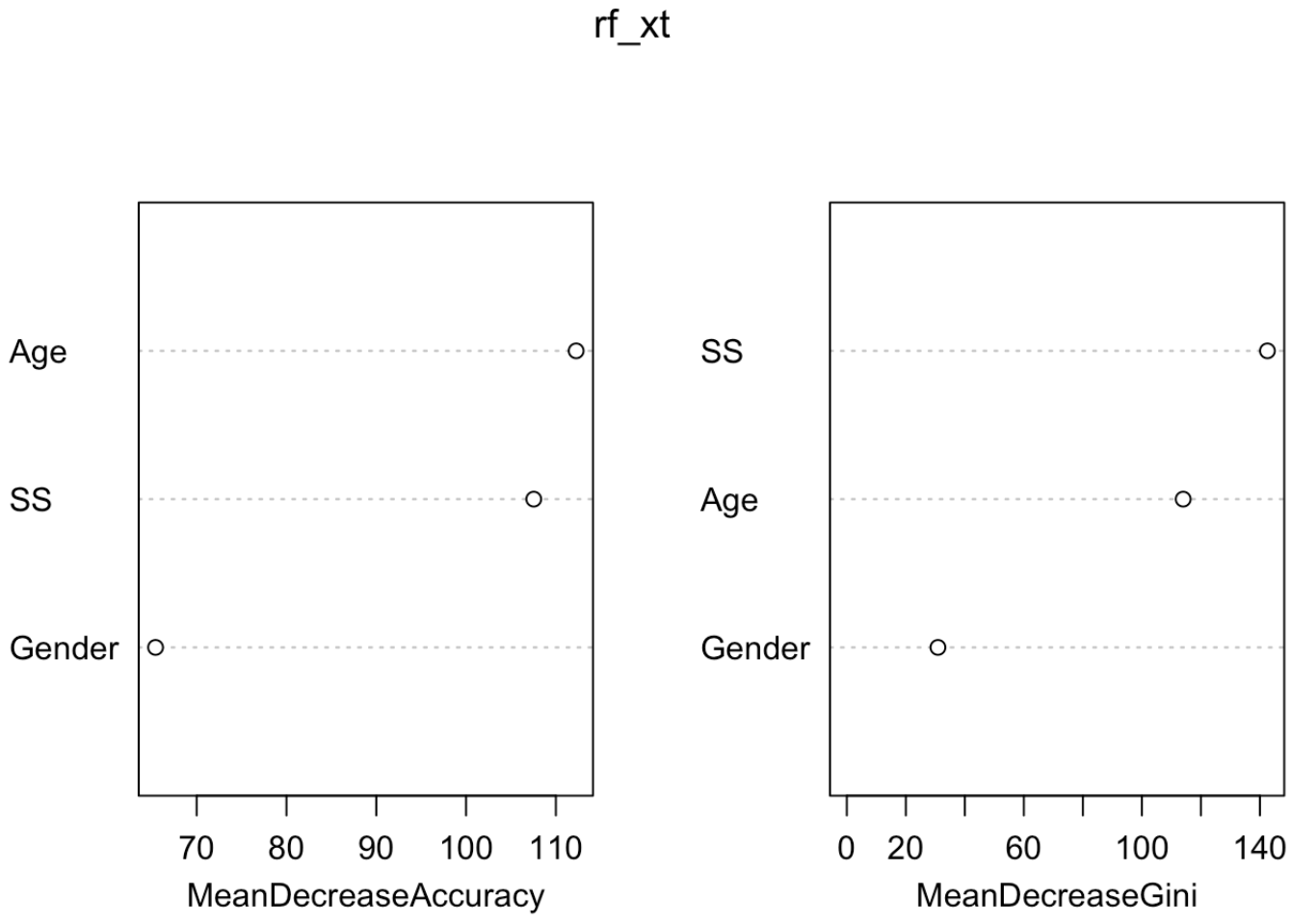

```
set.seed(365)

rf_xt <- randomForest(Ecstasy~.,
                      data = ecstasy,
                      importance = TRUE,
                      mtry = 2,
                      ntree = 1000,
                      CUTOFF = .6,
                      verbose = TRUE)
print(rf_xt)
```

```
##
## Call:
##  randomForest(formula = Ecstasy ~ ., data = ecstasy, importance = TRUE,      mtry
= 2, ntree = 1000, CUTOFF = 0.6, verbose = TRUE)
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 27.11%
## Confusion matrix:
##     0   1 class.error
## 0 872 262   0.2310406
## 1 249 502   0.3315579
```

```
f1(rf_xt) #Positive Class F1 Score function
```

```
## [1] "Precision: 0.6571, Recall: 0.6684, F1 Score: 0.6627"
```

```
varImpPlot(rf_xt)
```



```
set.seed(365)

rf_lh <- randomForest(Legalh~.,
                      data = legalh,
                      importance = TRUE,
                      mtry = 2,
                      ntree = 1000,
                      CUTOFF = .6,
                      verbose = TRUE)
print(rf_lh)
```

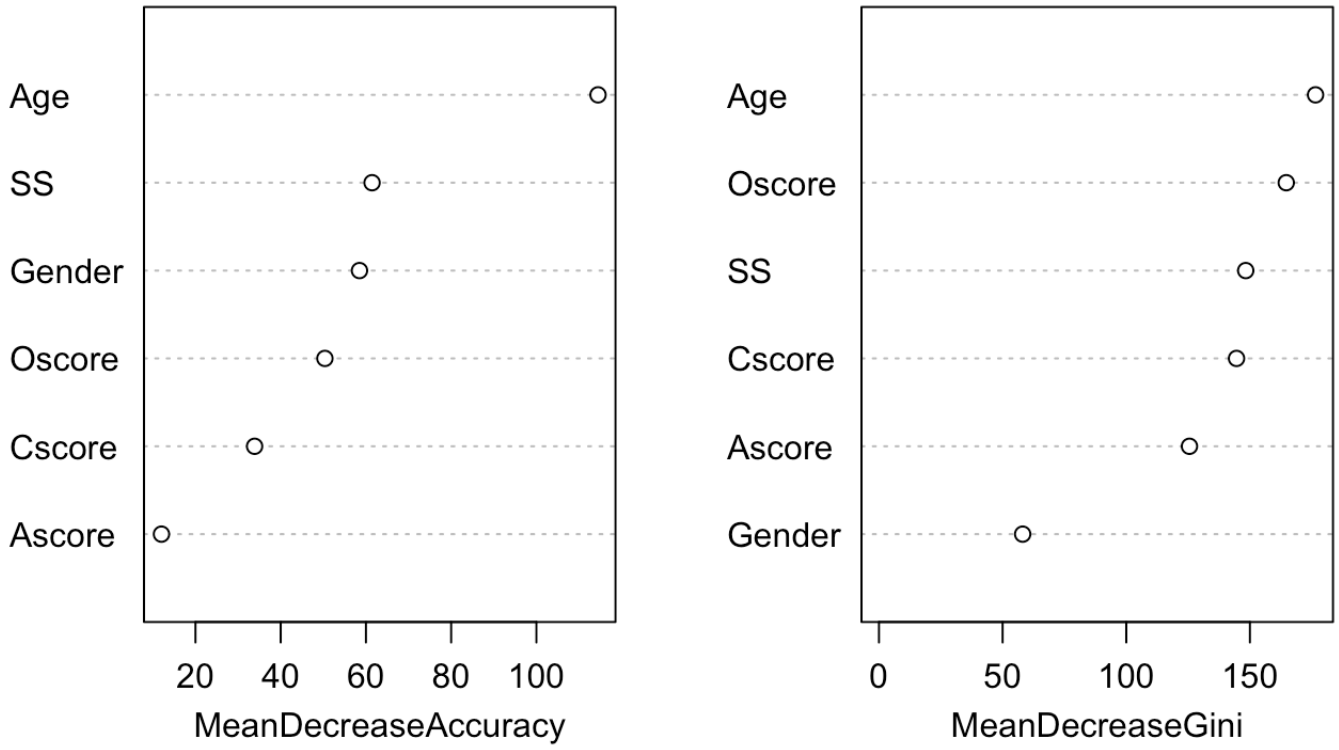
```
##
## Call:
##  randomForest(formula = Legalh ~ ., data = legalh, importance = TRUE,      mtry =
2, ntree = 1000, CUTOFF = 0.6, verbose = TRUE)
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 21.27%
## Confusion matrix:
##     0   1 class.error
## 0 939 184   0.1638468
## 1 217 545   0.2847769
```

```
f1(rf_lh) #Positive Class F1 Score function
```

```
## [1] "Precision: 0.7476, Recall: 0.7152, F1 Score: 0.7311"
```

```
varImpPlot(rf_lh)
```

rf_lh



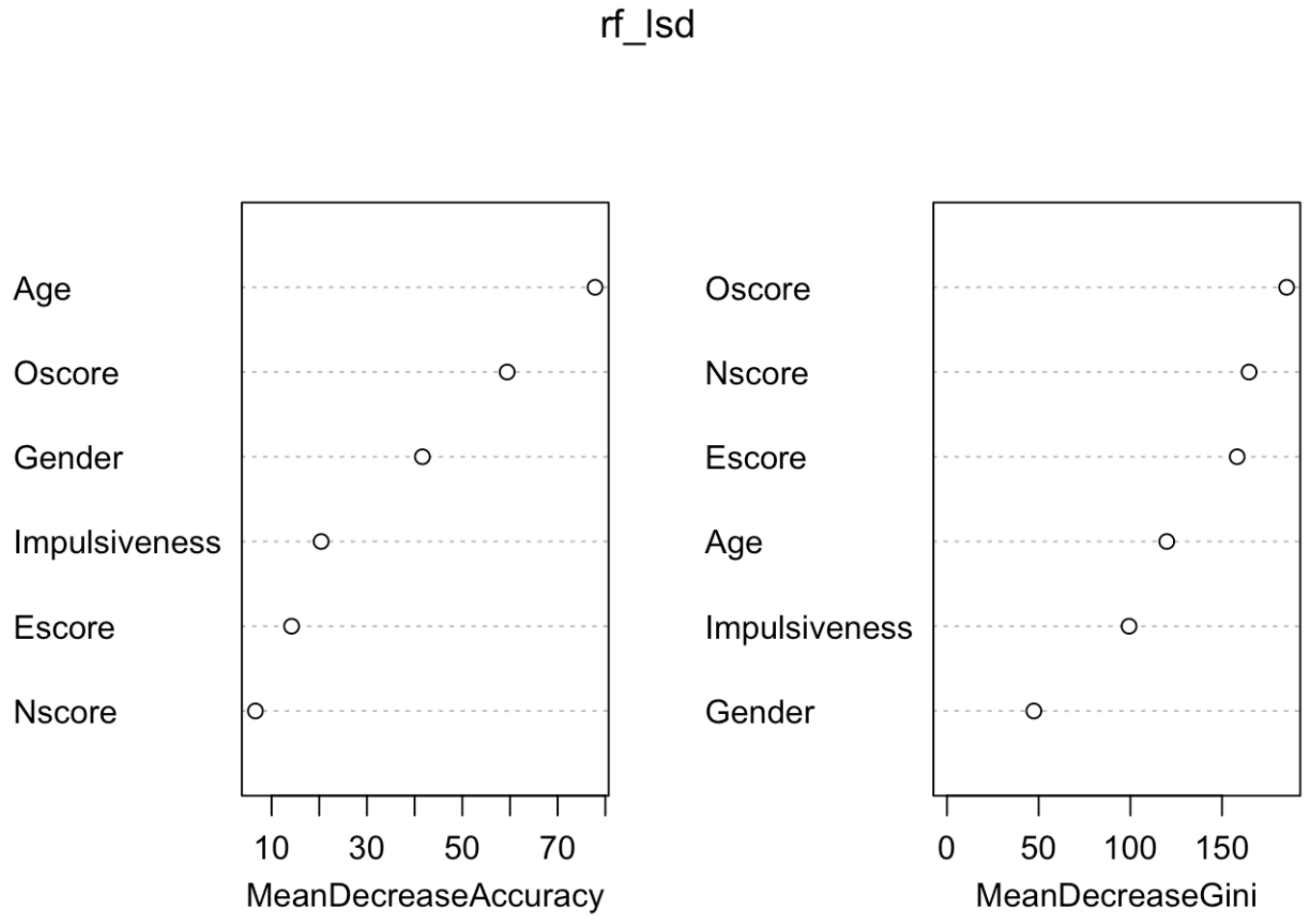

```
set.seed(365)

rf_lsd <- randomForest(LSD~.,
                       data = lsd,
                       importance = TRUE,
                       mtry = 3,
                       ntree = 1000,
                       CUTOFF = .6,
                       verbose = TRUE)
print(rf_lsd)
```

```
##
## Call:
##  randomForest(formula = LSD ~ ., data = lsd, importance = TRUE,      mtry = 3, ntr
ee = 1000, CUTOFF = 0.6, verbose = TRUE)
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 23.66%
## Confusion matrix:
##      0   1 class.error
## 0 1143 185   0.1393072
## 1  261 296   0.4685817
```

```
f1(rf_lsd) #Positive Class F1 Score function
```

```
## [1] "Precision: 0.6154, Recall: 0.5314, F1 Score: 0.5703"
```

```
varImpPlot(rf_lsd)
```



```
set.seed(365)

rf_nic <- randomForest(Nicotine~.,
                    data = nicotine,
                    importance = TRUE,
                    mtry = 2,
                    ntree = 1000,
                    CUTOFF = .6,
                    verbose = TRUE)
print(rf_nic)
```

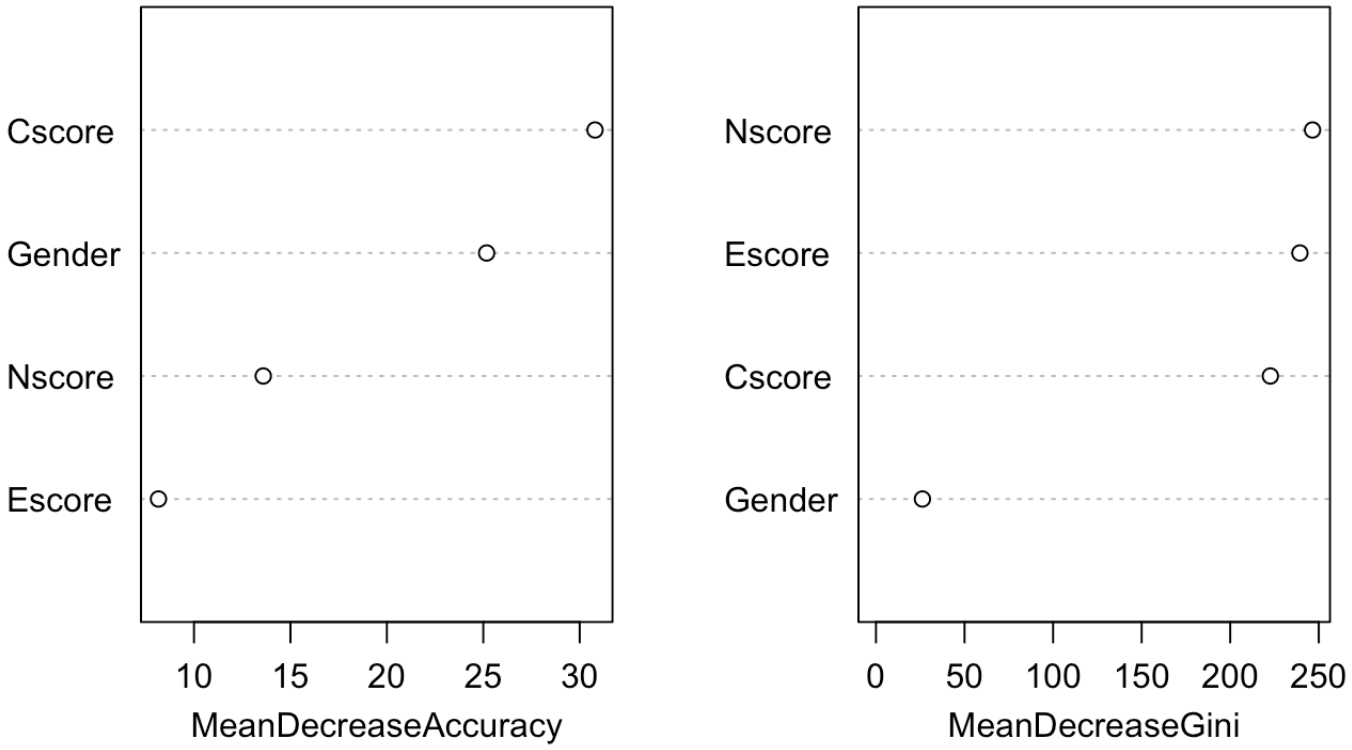
```
##
## Call:
##  randomForest(formula = Nicotine ~ ., data = nicotine, importance = TRUE,      mtr
y = 2, ntree = 1000, CUTOFF = 0.6, verbose = TRUE)
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 37.03%
## Confusion matrix:
##     0   1 class.error
## 0 189 432   0.6956522
## 1 266 998   0.2104430
```

```
f1(rf_nic) #Positive Class F1 Score function
```

```
## [1] "Precision: 0.6979, Recall: 0.7896, F1 Score: 0.7409"
```

```
varImpPlot(rf_nic)
```

rf_nic


Cscore
Gender
Nscore
Escore
10
15
20
25
30
MeanDecreaseAccuracy
Nscore
Escore
Cscore
Gender
0
50
100
150
200
250
MeanDecreaseGini

9.4 RFEX Code and Results

1. Importing the data and helper function(s)

```
library(readxl)
data <- read_excel("~/RStudio/CS_859_Team_Project/Dec1/Drug_Consumption_data_decade.x
lsx")

f_cols = c("Cannabis", "Ecstasy")

#drugs = list("Cannabis", "Ecstasy")

data[f_cols] <- lapply(data[f_cols], factor)

data
```

```
## # A tibble: 1,885 × 32
##       ID     Age Gender Education Country Ethni…¹ Nscore   Escore  Oscore Ascore
##    <dbl>   <dbl>  <dbl>     <dbl>   <dbl>   <dbl>  <dbl>    <dbl>   <dbl>  <dbl>
##  1     1  0.498   0.482   -0.0592   0.961   0.126  0.313 -0.575   -0.583  -0.917
##  2     2 -0.0785 -0.482    1.98     0.961  -0.317 -0.678  1.94     1.44    0.761
##  3     3  0.498  -0.482   -0.0592   0.961  -0.317 -0.467  0.805   -0.847  -1.62
##  4     4 -0.952   0.482    1.16     0.961  -0.317 -0.149 -0.806   -0.0193  0.590
##  5     5  0.498   0.482    1.98     0.961  -0.317  0.735 -1.63    -0.452  -0.302
##  6     6  2.59    0.482   -1.23     0.249  -0.317 -0.678 -0.300   -1.56    2.04
##  7     7  1.09   -0.482    1.16    -0.570  -0.317 -0.467 -1.09    -0.452  -0.302
##  8     8  0.498  -0.482   -1.74     0.961  -0.317 -1.33   1.94    -0.847  -0.302
##  9     9  0.498   0.482   -0.0592   0.249  -0.317  0.630  2.57    -0.976   0.761
## 10    10  1.82   -0.482    1.16     0.961  -0.317 -0.246  0.00332 -1.42    0.590
## # … with 1,875 more rows, 22 more variables: Cscore <dbl>, Impulsiveness <dbl>,
## #   SS <dbl>, Alcohol <dbl>, Amphet <dbl>, Amyl <dbl>, Benzos <dbl>,
## #   Caff <dbl>, Cannabis <fct>, Choc <dbl>, Coke <dbl>, Crack <dbl>,
## #   Ecstasy <fct>, Heroin <dbl>, Ketamine <dbl>, Legalh <dbl>, LSD <dbl>,
## #   Meth <dbl>, Mushrooms <dbl>, Nicotine <dbl>, VSA <dbl>, Label <dbl>, and
## #   abbreviated variable name ¹Ethnicity
```

```
# Feature Ranks for Cannabis -> 1:Age, 2:SS, 3:Oscore, 4:Cscore, 5:Nscore
# Feature Ranks for Ecstasy -> 1:Age, 2:SS, 3:Oscore, 4:Cscore, 5:Nscore

cannabis_top1 = data[c("Age", "Cannabis")]
ecstasy_top1 = data[c("Age", "Ecstasy")]

cannabis_top2 = data[c("Age", "SS", "Cannabis")]
ecstasy_top2 = data[c("Age", "SS", "Ecstasy")]

cannabis_top3 = data[c("Age", "Oscore", "SS", "Cannabis")]
ecstasy_top3 = data[c("Age", "Oscore", "SS", "Ecstasy")]

cannabis_top4 = data[c("Age", "Oscore", "Cscore", "SS", "Cannabis")]
ecstasy_top4 = data[c("Age", "Oscore", "Cscore", "SS", "Ecstasy")]

cannabis_top5 = data[c("Age", "Nscore", "Oscore", "Cscore", "SS", "Cannabis")]
ecstasy_top5 = data[c("Age", "Nscore", "Oscore", "Cscore", "SS", "Ecstasy")]
```

```
f1 <- function(rf_model)

{
p = rf_model$confusion[4]/(rf_model$confusion[3]+rf_model$confusion[4])
r = rf_model$confusion[4]/(rf_model$confusion[2]+rf_model$confusion[4])

score = (2*p*r)/(r+p)

sprintf('Precision: %.4f, Recall: %.4f, F1 Score: %.4f', p, r,score)
}
```

2. Top 1 Features

```
library(randomForest)
```

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
set.seed(365)

rf_can1 <- randomForest(Cannabis~.,
                    data = cannabis_top1,
                    importance = TRUE,
                    mtry = 1,
                    ntree = 1000,
                    CUTOFF = .6,
                    verbose = TRUE)
print(rf_can1)
```

```
##
## Call:
##  randomForest(formula = Cannabis ~ ., data = cannabis_top1, importance = TRUE,
mtry = 1, ntree = 1000, CUTOFF = 0.6, verbose = TRUE)
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 1
##
##         OOB estimate of  error rate: 27%
## Confusion matrix:
##     0    1 class.error
## 0 258  362   0.5838710
## 1 147 1118   0.1162055
```

```
f1(rf_can1) #Positive Class F1 Score function
```

```
## [1] "Precision: 0.7554, Recall: 0.8838, F1 Score: 0.8146"
```

```
#varImpPlot(rf_can1) #ranking invalid
```

```
set.seed(365)

rf_xt1 <- randomForest(Ecstasy~.,
                    data = ecstasy_top1,
                    importance = TRUE,
                    mtry = 1,
                    ntree = 1000,
                    CUTOFF = .6,
                    verbose = TRUE)
print(rf_xt1)
```

```
##
## Call:
##  randomForest(formula = Ecstasy ~ ., data = ecstasy_top1, importance = TRUE,
mtry = 1, ntree = 1000, CUTOFF = 0.6, verbose = TRUE)
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 1
##
##         OOB estimate of  error rate: 32.04%
## Confusion matrix:
##     0   1 class.error
## 0 886 248   0.2186949
## 1 356 395   0.4740346
```

```
f1(rf_xt1) #Positive Class F1 Score function
```

```
## [1] "Precision: 0.6143, Recall: 0.5260, F1 Score: 0.5667"
```

```
#varImpPlot(rf_xt1) #ranking invalid
```

3. Top 2 Features

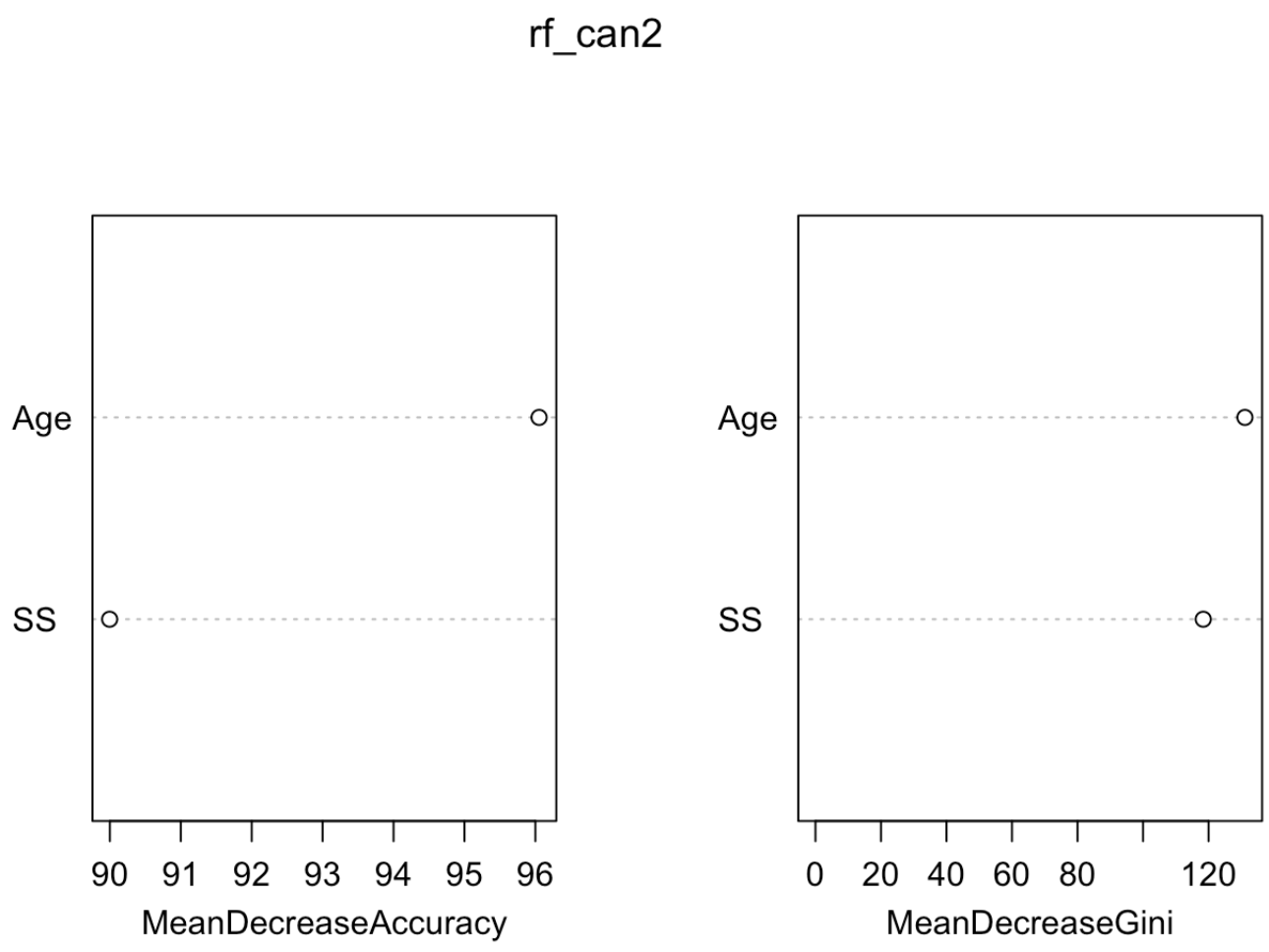
```
set.seed(365)

rf_can2 <- randomForest(Cannabis~.,
                    data = cannabis_top2,
                    importance = TRUE,
                    mtry = 1,
                    ntree = 1000,
                    CUTOFF = .6,
                    verbose = TRUE)
print(rf_can2)
```

```
##
## Call:
##  randomForest(formula = Cannabis ~ ., data = cannabis_top2, importance = TRUE,
mtry = 1, ntree = 1000, CUTOFF = 0.6, verbose = TRUE)
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 1
##
##         OOB estimate of  error rate: 22.81%
## Confusion matrix:
##     0    1 class.error
## 0 321  299   0.4822581
## 1 131 1134   0.1035573
```

```
f1(rf_can2) #Positive Class F1 Score function
```

```
## [1] "Precision: 0.7913, Recall: 0.8964, F1 Score: 0.8406"
```

```
varImpPlot(rf_can2)
```



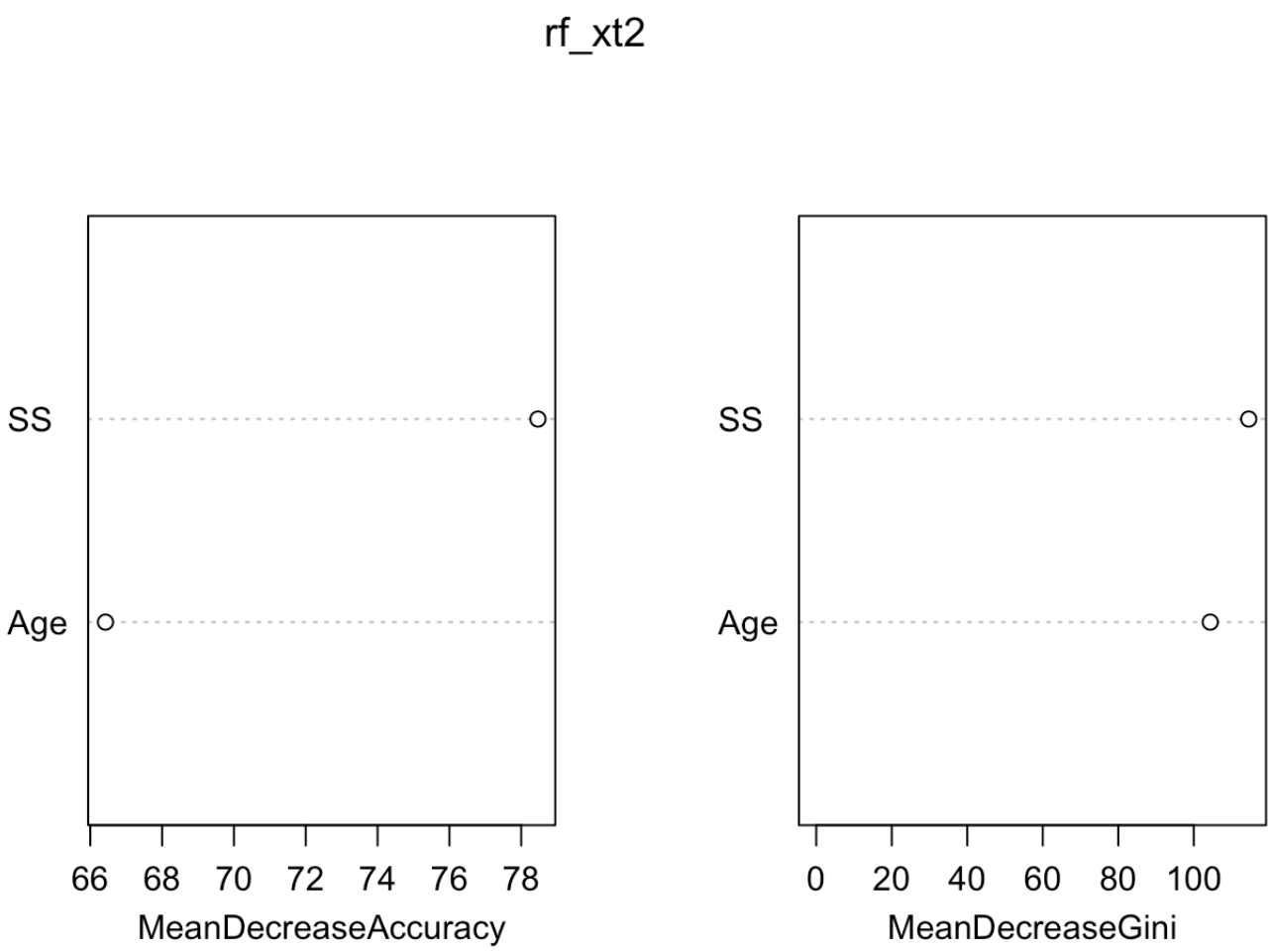
```
set.seed(365)

rf_xt2 <- randomForest(Ecstasy~.,
                     data = ecstasy_top2,
                     importance = TRUE,
                     mtry = 1,
                     ntree = 1000,
                     CUTOFF = .6,
                     verbose = TRUE)
print(rf_xt2)
```

```
##
## Call:
##  randomForest(formula = Ecstasy ~ ., data = ecstasy_top2, importance = TRUE,
mtry = 1, ntree = 1000, CUTOFF = 0.6, verbose = TRUE)
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 1
##
##         OOB estimate of  error rate: 29.6%
## Confusion matrix:
##     0   1 class.error
## 0 840 294   0.2592593
## 1 264 487   0.3515313
```

```
f1(rf_xt2) #Positive Class F1 Score function
```

```
## [1] "Precision: 0.6236, Recall: 0.6485, F1 Score: 0.6358"
```

```
varImpPlot(rf_xt2)
```



4. Top 3 Features

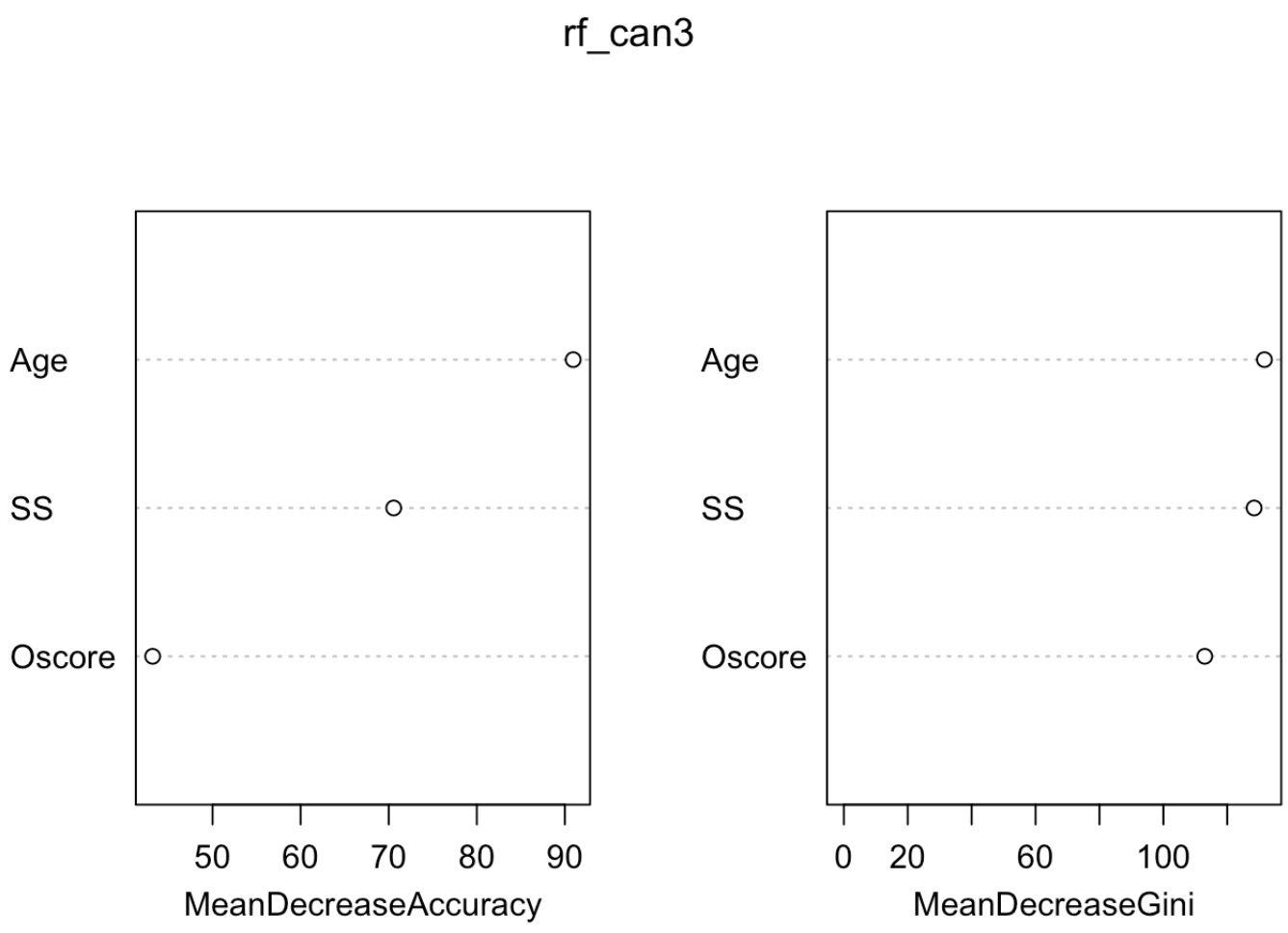
```
set.seed(365)

rf_can3 <- randomForest(Cannabis~.,
                    data = cannabis_top3,
                    importance = TRUE,
                    mtry = 1,
                    ntree = 1000,
                    CUTOFF = .6,
                    verbose = TRUE)
print(rf_can3)
```

```
##
## Call:
##  randomForest(formula = Cannabis ~ ., data = cannabis_top3, importance = TRUE,
mtry = 1, ntree = 1000, CUTOFF = 0.6, verbose = TRUE)
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 1
##
##         OOB estimate of  error rate: 23.24%
## Confusion matrix:
##     0    1 class.error
## 0 349  271   0.4370968
## 1 167 1098   0.1320158
```

```
f1(rf_can3) #Positive Class F1 Score function
```

```
## [1] "Precision: 0.8020, Recall: 0.8680, F1 Score: 0.8337"
```

```
varImpPlot(rf_can3)
```



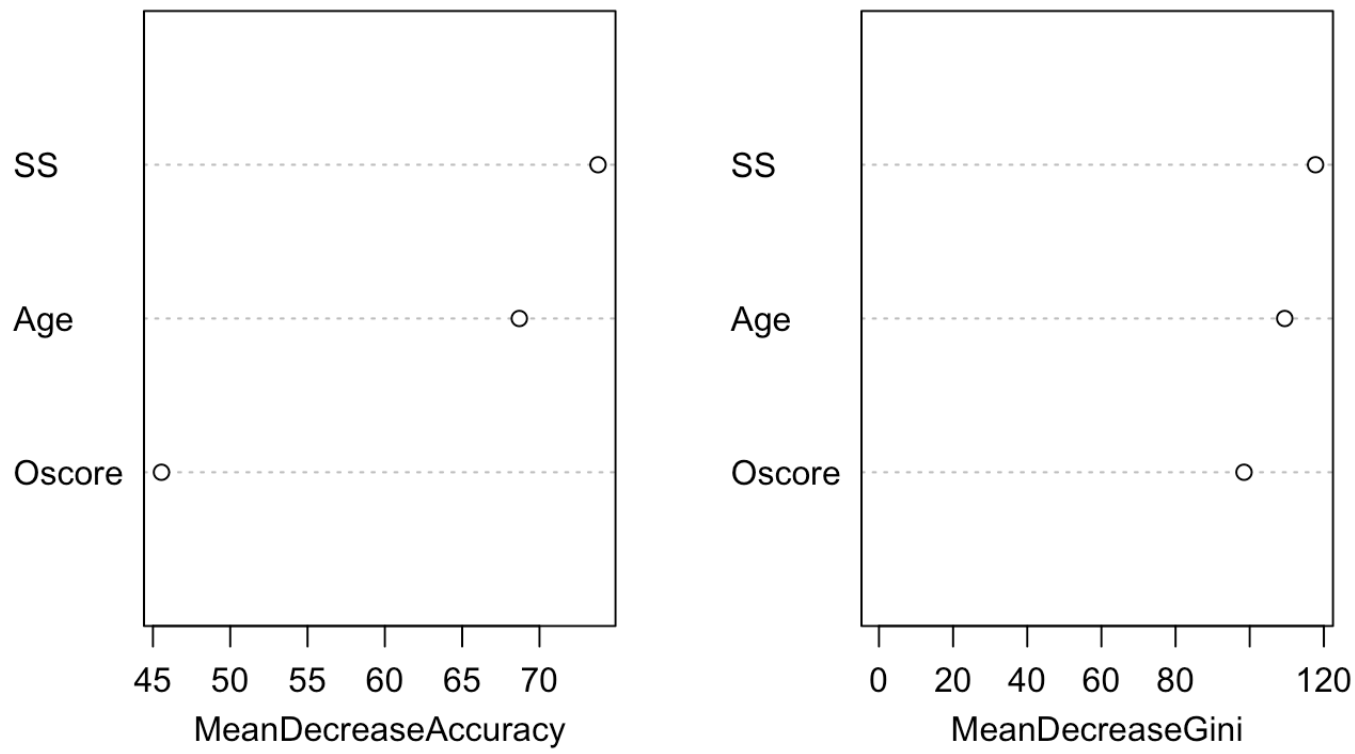
```
set.seed(365)

rf_xt3 <- randomForest(Ecstasy~.,
                     data = ecstasy_top3,
                     importance = TRUE,
                     mtry = 1,
                     ntree = 1000,
                     CUTOFF = .6,
                     verbose = TRUE)
print(rf_xt3)
```

```
##
## Call:
##  randomForest(formula = Ecstasy ~ ., data = ecstasy_top3, importance = TRUE,
mtry = 1, ntree = 1000, CUTOFF = 0.6, verbose = TRUE)
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 1
##
##         OOB estimate of  error rate: 27.59%
## Confusion matrix:
##     0   1 class.error
## 0 864 270   0.2380952
## 1 250 501   0.3328895
```

```
f1(rf_xt3) #Positive Class F1 Score function
```

```
## [1] "Precision: 0.6498, Recall: 0.6671, F1 Score: 0.6583"
```

```
varImpPlot(rf_xt3)
```



5. Top 4 Features

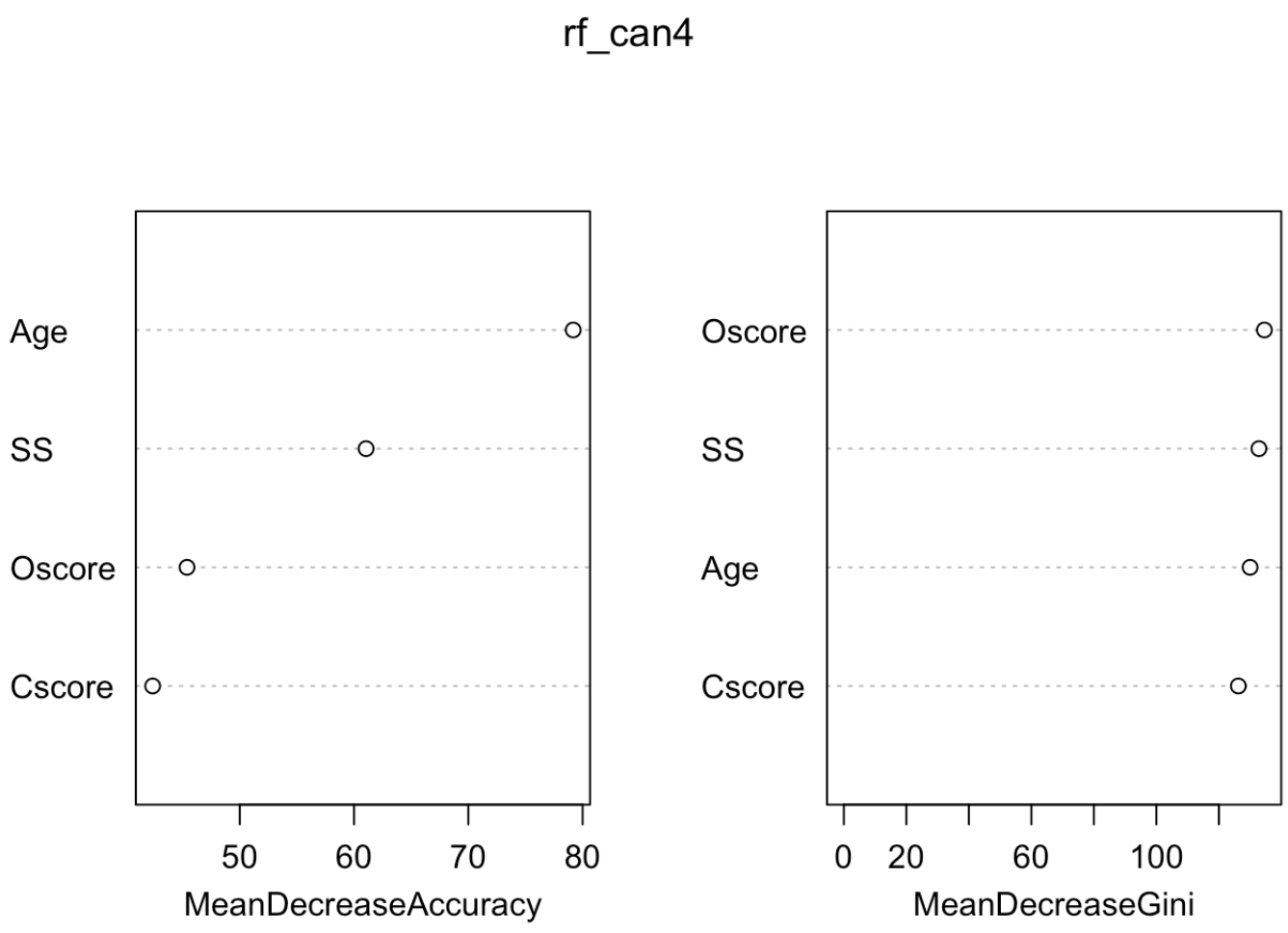
```
set.seed(365)

rf_can4 <- randomForest(Cannabis~.,
                    data = cannabis_top4,
                    importance = TRUE,
                    mtry = 1,
                    ntree = 1000,
                    CUTOFF = .6,
                    verbose = TRUE)
print(rf_can4)
```

```
##
## Call:
##  randomForest(formula = Cannabis ~ ., data = cannabis_top4, importance = TRUE,
mtry = 1, ntree = 1000, CUTOFF = 0.6, verbose = TRUE)
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 1
##
##         OOB estimate of  error rate: 22.44%
## Confusion matrix:
##     0    1 class.error
## 0 371  249   0.4016129
## 1 174 1091   0.1375494
```

```
f1(rf_can4) #Positive Class F1 Score function
```

```
## [1] "Precision: 0.8142, Recall: 0.8625, F1 Score: 0.8376"
```

```
varImpPlot(rf_can4)
```



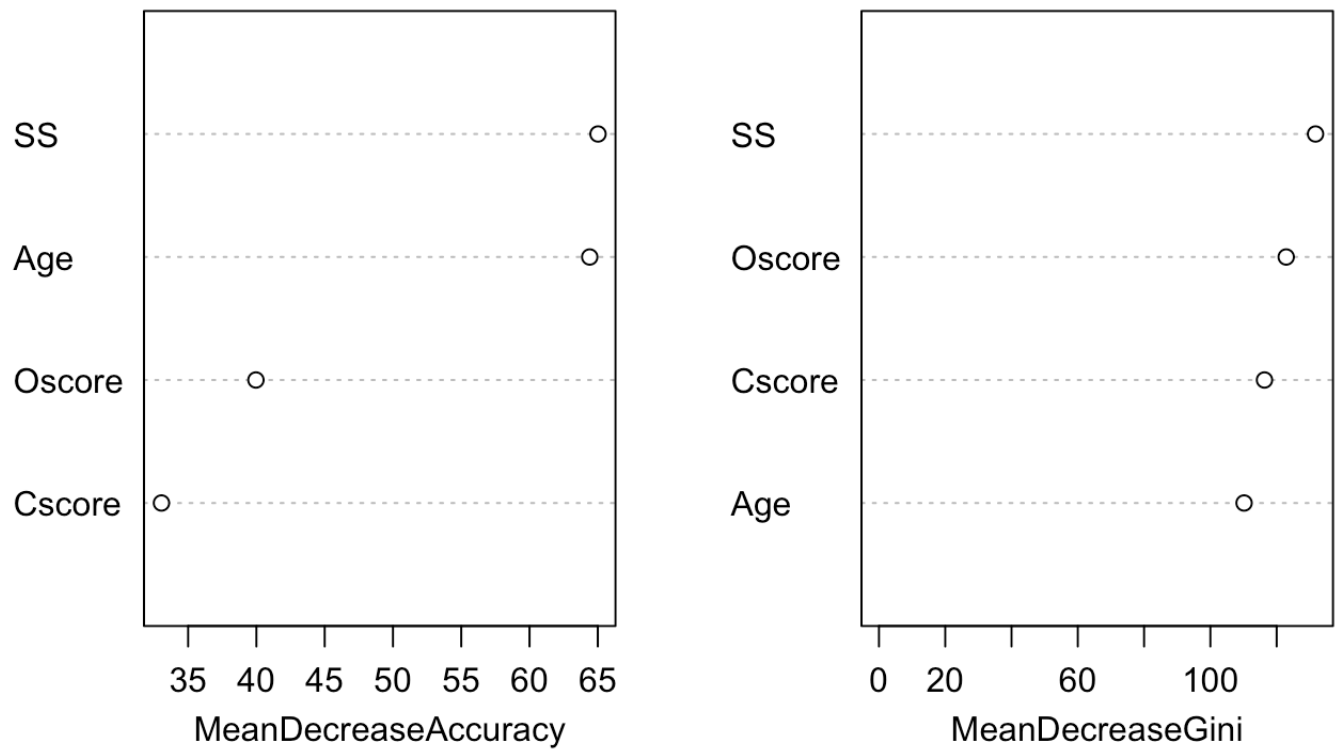
```
set.seed(365)

rf_xt4 <- randomForest(Ecstasy~.,
                    data = ecstasy_top4,
                    importance = TRUE,
                    mtry = 1,
                    ntree = 1000,
                    CUTOFF = .6,
                    verbose = TRUE)
print(rf_xt4)
```

```
##
## Call:
##  randomForest(formula = Ecstasy ~ ., data = ecstasy_top4, importance = TRUE,
mtry = 1, ntree = 1000, CUTOFF = 0.6, verbose = TRUE)
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 1
##
##         OOB estimate of  error rate: 27.37%
## Confusion matrix:
##     0   1 class.error
## 0 868 266   0.2345679
## 1 250 501   0.3328895
```

```
f1(rf_xt4) #Positive Class F1 Score function
```

```
## [1] "Precision: 0.6532, Recall: 0.6671, F1 Score: 0.6601"
```

```
varImpPlot(rf_xt4)
```



6. Top 5 Features

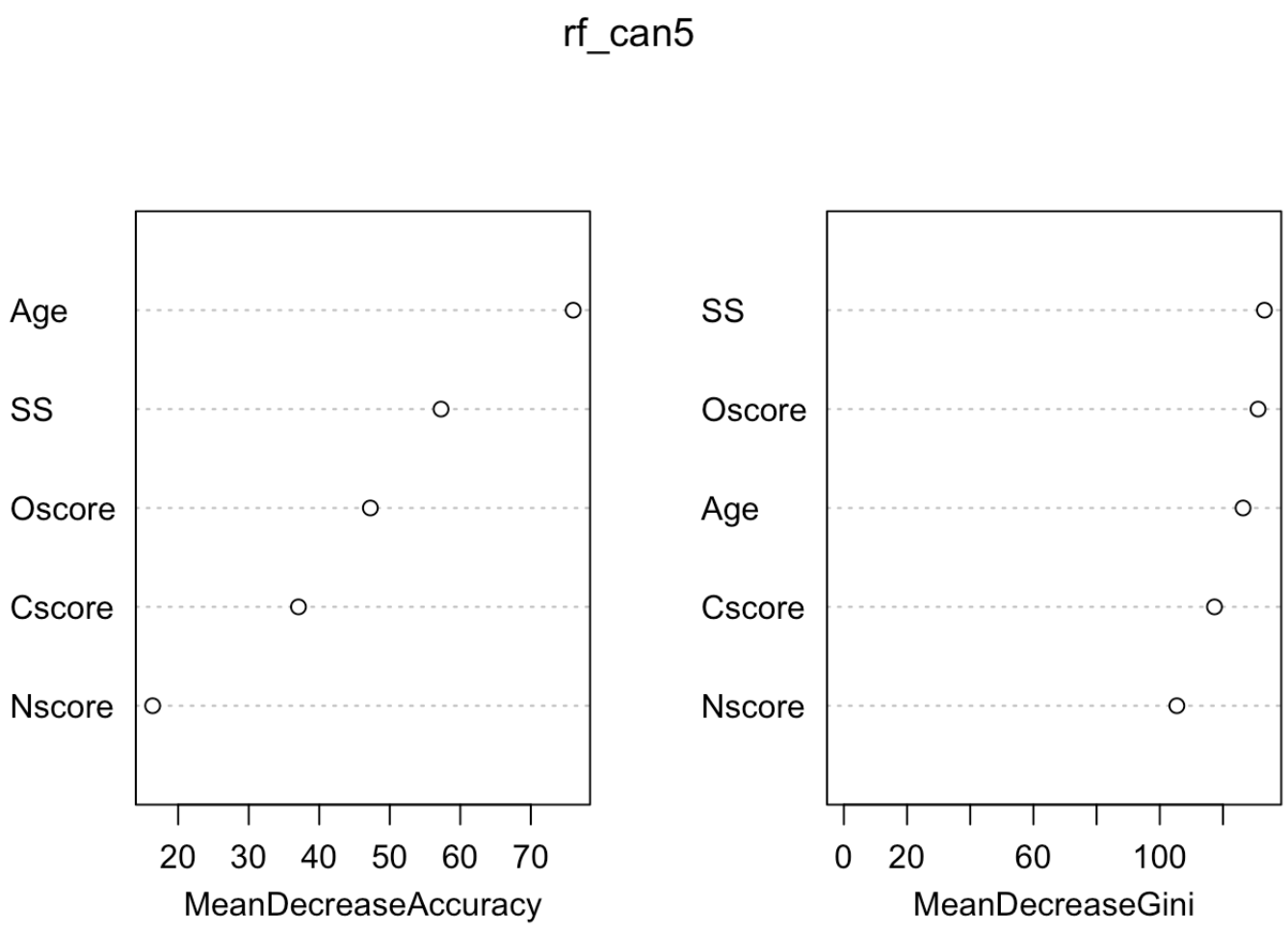
```
set.seed(365)

rf_can5 <- randomForest(Cannabis~.,
                    data = cannabis_top5,
                    importance = TRUE,
                    mtry = 1,
                    ntree = 1000,
                    CUTOFF = .6,
                    verbose = TRUE)
print(rf_can5)
```

```
##
## Call:
##  randomForest(formula = Cannabis ~ ., data = cannabis_top5, importance = TRUE,
mtry = 1, ntree = 1000, CUTOFF = 0.6, verbose = TRUE)
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 1
##
##         OOB estimate of  error rate: 21.06%
## Confusion matrix:
##     0    1 class.error
## 0 380  240   0.3870968
## 1 157 1108   0.1241107
```

```
f1(rf_can5) #Positive Class F1 Score function
```

```
## [1] "Precision: 0.8220, Recall: 0.8759, F1 Score: 0.8481"
```

```
varImpPlot(rf_can5)
```



```
set.seed(365)

rf_xt5 <- randomForest(Ecstasy~.,
                    data = ecstasy_top5,
                    importance = TRUE,
                    mtry = 1,
                    ntree = 1000,
                    CUTOFF = .6,
                    verbose = TRUE)
print(rf_xt5)
```

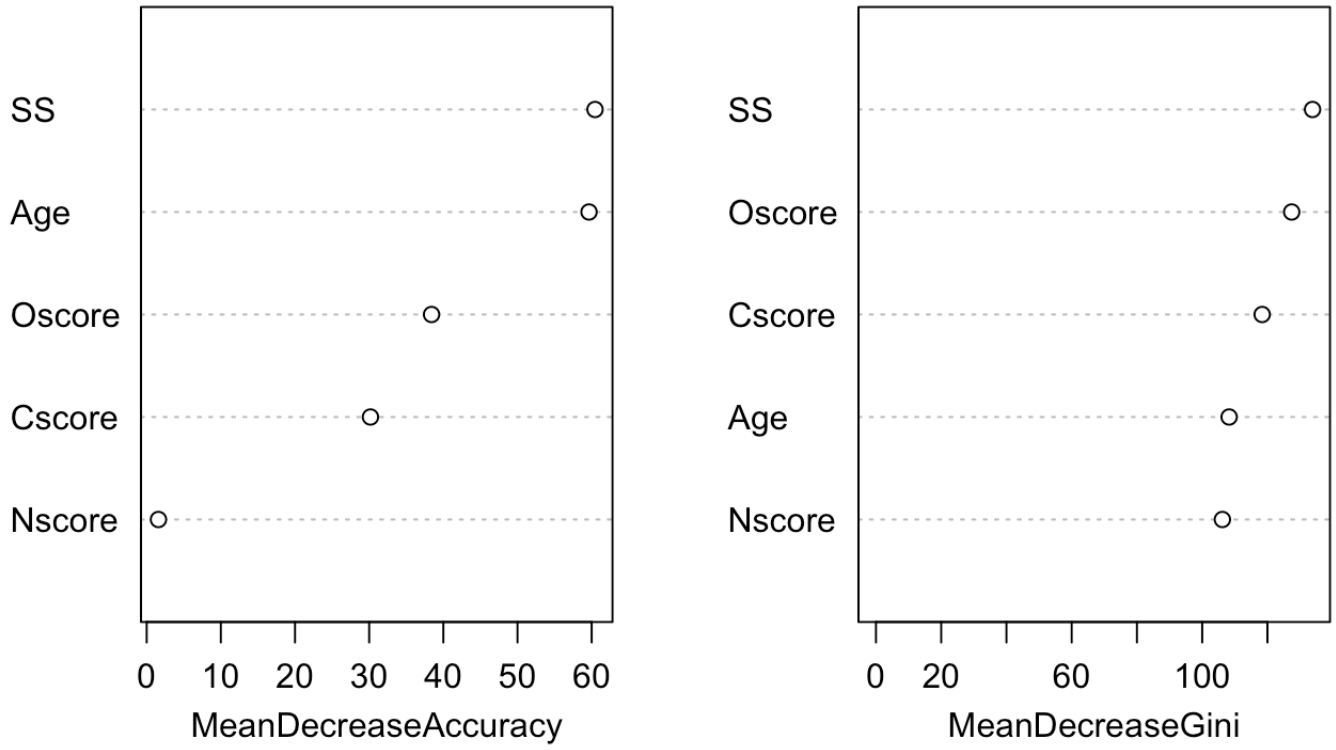
```
##
## Call:
##  randomForest(formula = Ecstasy ~ ., data = ecstasy_top5, importance = TRUE,
mtry = 1, ntree = 1000, CUTOFF = 0.6, verbose = TRUE)
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 1
##
##         OOB estimate of  error rate: 28.12%
## Confusion matrix:
##     0   1 class.error
## 0 871 263   0.2319224
## 1 267 484   0.3555260
```

```
f1(rf_xt5) #Positive Class F1 Score function
```

```
## [1] "Precision: 0.6479, Recall: 0.6445, F1 Score: 0.6462"
```

```
varImpPlot(rf_xt5)
```

rf_xt5


SS
Age
Oscore
Cscore
Nscore
0
10
20
30
40
50
60
MeanDecreaseAccuracy
SS
Oscore
Cscore
Age
Nscore
0
20
60
100
MeanDecreaseGini

9.5 Rebalanced and Feature Based Replication Data

```
#load libraries
library(randomForest) #for random forest
library(caret) #for partitioning data
library(ROCR) #for AUC/ROC plotting
library(dplyr) #for selecting random samples

#optima
ntreeOPT <- 1000 #found from lawrence's grid search
mtryOPT <- 4 #
CUTOFFOPT <- 0.6 #

#load data from working directory
drugMonth <- read.csv("drugConsumptionDataMonth.csv", header=TRUE)
drugMonth$Label <- as.factor(drugMonth$Label) #changes label data of last
column to factor
#to allow classification


#Cocaine
cokeMonth <- filter(drugMonth, Coke==1)
cokeMonth$Coke <- as.factor(cokeMonth$Coke) #changes label data of
relevant column to factor
#to allow classification
cokeMonth = cokeMonth[c('Age', 'Gender', 'Education', 'Nscore', 'Escore',
'Oscore', 'Ascore',
                        'Cscore', 'Impulsiveness','SS', 'Coke')]
noCokeMonth <- filter(drugMonth, Coke==0)
noCokeMonth$Coke <- as.factor(noCokeMonth$Coke) #changes label data of
relevant column to factor
#to allow classification
noCokeMonth = noCokeMonth[c('Age', 'Gender', 'Education', 'Nscore',
'Escore', 'Oscore', 'Ascore',
                        'Cscore', 'Impulsiveness','SS', 'Coke')]

#to get 2:1 for neg:pos class proportions
noCokeMonthBalanced <- sample_frac(noCokeMonth, 0.18)

cokeTrainMonth <- rbind(cokeMonth, noCokeMonthBalanced)

#produces a random forest model named model1.rf
cokeModelMonth.rf <- randomForest(Coke ~., data = cokeTrainMonth,
                          ntree = ntreeOPT, #how many trees
                          mtry = mtryOPT,
                          CUTOFF = CUTOFFOPT, #how many have to agree for
cutoff
                          importance = TRUE)

#print(paste("optimal number of trees: ", ntreeOpt))
#print(paste("optimal number of variables tried at each node: ", mtryOpt))
#print(paste("optimal cutoff proportion: ", CUTOFFOPT))
#print(paste("lowest OOB: ",lowestOOB))
print(cokeModelMonth.rf) #display OOB, confusion matrix, and so on for
model1.rf

importance(cokeModelMonth.rf)
varImpPlot(cokeModelMonth.rf)

#LSD
```

```
LSDMonth <- filter(drugMonth, LSD==1)
LSDMonth$LSD <- as.factor(LSDMonth$LSD) #changes label data of relevant
column to factor
#to allow classification
LSDMonth = LSDMonth[c('Age', 'Gender', 'Education', 'Nscore', 'Escore',
'Oscore', 'Ascore',
                      'Cscore', 'Impulsiveness','SS', 'LSD')]
noLSDMonth <- filter(drugMonth, LSD==0)
noLSDMonth$LSD <- as.factor(noLSDMonth$LSD) #changes label data of
relevant column to factor
#to allow classification
noLSDMonth = noLSDMonth[c('Age', 'Gender', 'Education', 'Nscore',
'Escore', 'Oscore', 'Ascore',
                      'Cscore', 'Impulsiveness','SS', 'LSD')]

#to get 2:1 for neg:pos class proportions
noLSDMonthBalanced <- sample_frac(noLSDMonth, 0.19)

LSDTrainMonth <- rbind(LSDMonth, noLSDMonthBalanced)

#produces a random forest model named model1.rf
LSDModelMonth.rf <- randomForest(LSD ~., data = LSDTrainMonth,
                         ntree = ntreeOPT, #how many trees
                         mtry = mtryOPT,
                         CUTOFF = CUTOFFOPT, #how many have to agree for
cutoff
                         importance = TRUE)

#print(paste("optimal number of trees: ", ntreeOpt))
#print(paste("optimal number of variables tried at each node: ", mtryOpt))
#print(paste("optimal cutoff proportion: ", CUTOFFOPT))
#print(paste("lowest OOB: ",lowestOOB))
print(LSDModelMonth.rf) #display OOB, confusion matrix, and so on for
model1.rf

importance(LSDModelMonth.rf)
varImpPlot(LSDModelMonth.rf)

#Mushrooms
MushroomsMonth <- filter(drugMonth, Mushrooms==1)
MushroomsMonth$Mushrooms <- as.factor(MushroomsMonth$Mushrooms) #changes
label data of relevant column to factor
#to allow classification
MushroomsMonth = MushroomsMonth[c('Age', 'Gender', 'Education', 'Nscore',
'Escore', 'Oscore', 'Ascore',
                      'Cscore', 'Impulsiveness','SS', 'Mushrooms')]
noMushroomsMonth <- filter(drugMonth, Mushrooms==0)
noMushroomsMonth$Mushrooms <- as.factor(noMushroomsMonth$Mushrooms)
#changes label data of last column to factor
#to allow classification
noMushroomsMonth = noMushroomsMonth[c('Age', 'Gender', 'Education',
'Nscore', 'Escore', 'Oscore', 'Ascore',
                      'Cscore', 'Impulsiveness','SS', 'Mushrooms')]

#to get 2:1 for neg:pos class proportions
noMushroomsMonthBalanced <- sample_frac(noMushroomsMonth, 0.18)

MushroomsTrainMonth <- rbind(MushroomsMonth, noMushroomsMonthBalanced)
```

```
#produces a random forest model named model1.rf
MushroomsModelMonth.rf <- randomForest(Mushrooms ~., data =
MushroomsTrainMonth,
                          ntree = ntreeOPT, #how many trees
                          mtry = mtryOPT,
                          CUTOFF = CUTOFFOPT, #how many have to agree for
cutoff
                          importance = TRUE)

#print(paste("optimal number of trees: ", ntreeOpt))
#print(paste("optimal number of variables tried at each node: ", mtryOpt))
#print(paste("optimal cutoff proportion: ", CUTOFFOPT))
#print(paste("lowest OOB: ",lowestOOB))
print(MushroomsModelMonth.rf) #display OOB, confusion matrix, and so on
for model1.rf

importance(MushroomsModelMonth.rf)
varImpPlot(MushroomsModelMonth.rf)

#Methadone
MethMonth <- filter(drugMonth, Meth==1)
MethMonth$Meth <- as.factor(MethMonth$Meth) #changes label data of
relevant column to factor
#to allow classification
MethMonth = MethMonth[c('Age', 'Gender', 'Education', 'Nscore', 'Escore',
'Oscore', 'Ascore',
                        'Cscore', 'Impulsiveness','SS', 'Meth')]
noMethMonth <- filter(drugMonth, Meth==0)
noMethMonth$Meth <- as.factor(noMethMonth$Meth) #changes label data of
last column to factor
#to allow classification
noMethMonth = noMethMonth[c('Age', 'Gender', 'Education', 'Nscore',
'Escore', 'Oscore', 'Ascore',
                        'Cscore', 'Impulsiveness','SS', 'Meth')]

#to get 2:1 for neg:pos class proportions
noMethMonthBalanced <- sample_frac(noMethMonth, 0.19)

MethTrainMonth <- rbind(MethMonth, noMethMonthBalanced)

#produces a random forest model named model1.rf
MethModelMonth.rf <- randomForest(Meth ~., data = MethTrainMonth,
                          ntree = ntreeOPT, #how many trees
                          mtry = mtryOPT,
                          CUTOFF = CUTOFFOPT, #how many have to agree for
cutoff
                          importance = TRUE)

#print(paste("optimal number of trees: ", ntreeOpt))
#print(paste("optimal number of variables tried at each node: ", mtryOpt))
#print(paste("optimal cutoff proportion: ", CUTOFFOPT))
#print(paste("lowest OOB: ",lowestOOB))
print(MethModelMonth.rf) #display OOB, confusion matrix, and so on for
model1.rf

importance(MethModelMonth.rf)
varImpPlot(MethModelMonth.rf)
```

```
#Crack
CrackMonth <- filter(drugMonth, Crack==1)
CrackMonth$Crack <- as.factor(CrackMonth$Crack) #changes label data of
relevant column to factor
#to allow classification
CrackMonth = CrackMonth[c('Age', 'Gender', 'Education', 'Nscore',
'Escore', 'Oscore', 'Ascore',
                        'Cscore', 'Impulsiveness','SS', 'Crack')]
noCrackMonth <- filter(drugMonth, Crack==0)
noCrackMonth$Crack <- as.factor(noCrackMonth$Crack) #changes label data of
last column to factor
#to allow classification
noCrackMonth = noCrackMonth[c('Age', 'Gender', 'Education', 'Nscore',
'Escore', 'Oscore', 'Ascore',
                        'Cscore', 'Impulsiveness','SS', 'Crack')]

#to get 2:1 for neg:pos class proportions
noCrackMonthBalanced <- sample_frac(noCrackMonth, 0.02)

CrackTrainMonth <- rbind(CrackMonth, noCrackMonthBalanced)

#produces a random forest model named model1.rf
CrackModelMonth.rf <- randomForest(Crack ~., data = CrackTrainMonth,
                          ntree = ntreeOPT, #how many trees
                          mtry = mtryOPT,
                          CUTOFF = CUTOFFOPT, #how many have to agree for
cutoff
                          importance = TRUE)

#print(paste("optimal number of trees: ", ntreeOpt))
#print(paste("optimal number of variables tried at each node: ", mtryOpt))
#print(paste("optimal cutoff proportion: ", CUTOFFOPT))
#print(paste("lowest OOB: ",lowestOOB))
print(CrackModelMonth.rf) #display OOB, confusion matrix, and so on for
model1.rf

importance(CrackModelMonth.rf)
varImpPlot(CrackModelMonth.rf)

#Heroin
HeroinMonth <- filter(drugMonth, Heroin==1)
HeroinMonth$Heroin <- as.factor(HeroinMonth$Heroin) #changes label data of
relevant column to factor
#to allow classification
HeroinMonth = HeroinMonth[c('Age', 'Gender', 'Education', 'Nscore',
'Escore', 'Oscore', 'Ascore',
                        'Cscore', 'Impulsiveness','SS', 'Heroin')]
noHeroinMonth <- filter(drugMonth, Heroin==0)
noHeroinMonth$Heroin <- as.factor(noHeroinMonth$Heroin) #changes label
data of last column to factor
#to allow classification
noHeroinMonth = noHeroinMonth[c('Age', 'Gender', 'Education', 'Nscore',
'Escore', 'Oscore', 'Ascore',
                        'Cscore', 'Impulsiveness','SS', 'Heroin')]

#to get 2:1 for neg:pos class proportions
noHeroinMonthBalanced <- sample_frac(noHeroinMonth, 0.05)
```

```
HeroinTrainMonth <- rbind(HeroinMonth, noHeroinMonthBalanced)

#produces a random forest model named model1.rf
HeroinModelMonth.rf <- randomForest(Heroin ~., data = HeroinTrainMonth,
                          ntree = ntreeOPT, #how many trees
                          mtry = mtryOPT,
                          CUTOFF = CUTOFFOPT, #how many have to agree for
cutoff
                          importance = TRUE)

#print(paste("optimal number of trees: ", ntreeOpt))
#print(paste("optimal number of variables tried at each node: ", mtryOpt))
#print(paste("optimal cutoff proportion: ", CUTOFFOPT))
#print(paste("lowest OOB: ",lowestOOB))
print(HeroinModelMonth.rf) #display OOB, confusion matrix, and so on for
model1.rf

importance(HeroinModelMonth.rf)
varImpPlot(HeroinModelMonth.rf)



#CORROBORATION CODE

#load data from working directory
drugDecade <- read.csv("DrugConsumptionDataDecade.csv", header=TRUE)
drugDecade$Label <- as.factor(drugDecade$Label) #changes label data of
last column to factor
#to allow classification

#Crack
CrackDecade <- filter(drugDecade, Crack==1)
CrackDecade$Crack <- as.factor(CrackDecade$Crack) #changes label data of
relevant column to factor
#to allow classification
#all the features
CrackDecade = CrackDecade[c('Age', 'Gender', 'Education', 'Nscore',
'Escore', 'Oscore', 'Ascore',
                           'Cscore', 'Impulsiveness','SS', 'Crack')]
#only the paper features (table 18)
#CrackDecade = CrackDecade[c('Escore', 'Cscore', 'Crack')]
noCrackDecade <- filter(drugDecade, Crack==0)
noCrackDecade$Crack <- as.factor(noCrackDecade$Crack) #changes label data
of last column to factor
#to allow classification
#all features
noCrackDecade = noCrackDecade[c('Age', 'Gender', 'Education', 'Nscore',
'Escore', 'Oscore', 'Ascore',
                               'Cscore', 'Impulsiveness','SS', 'Crack')]
#just table 18 features
#noCrackDecade = noCrackDecade[c('Escore','Cscore', 'Crack')]

#to get 2:1 for neg:pos class proportions
noCrackDecadeBalanced <- sample_frac(noCrackDecade, 1)

CrackTrainDecade <- rbind(CrackDecade, noCrackDecadeBalanced)
```

```
#produces a random forest model named model1.rf
CrackModelDecade.rf <- randomForest(Crack ~., data = CrackTrainDecade,
                                    ntree = ntreeOPT, #how many trees
                                    mtry = mtryOPT,
                                    CUTOFF = CUTOFFOPT, #how many have to
agree for cutoff
                                    importance = TRUE)

#print(paste("optimal number of trees: ", ntreeOpt))
#print(paste("optimal number of variables tried at each node: ", mtryOpt))
#print(paste("optimal cutoff proportion: ", CUTOFFOPT))
#print(paste("lowest OOB: ",lowestOOB))
print(CrackModelDecade.rf) #display OOB, confusion matrix, and so on for
model1.rf

importance(CrackModelDecade.rf)
varImpPlot(CrackModelDecade.rf)

#LSD CORROBORATION
LSDDecade <- filter(drugDecade, LSD==1)
LSDDecade$LSD <- as.factor(LSDDecade$LSD) #changes label data of relevant
column to factor
#to allow classification
#all the features
LSDDecade = LSDDecade[c('Age', 'Gender', 'Education', 'Nscore', 'Escore',
'Oscore', 'Ascore',
                        'Cscore', 'Impulsiveness','SS', 'LSD')]
#only paper features (table 18)
#LSDDecade = LSDDecade[c('Age', 'Gender', 'Nscore', 'Escore', 'Oscore',
'Impulsiveness', 'LSD')]
noLSDDecade <- filter(drugDecade, LSD==0)
noLSDDecade$LSD <- as.factor(noLSDDecade$LSD) #changes label data of
relevant column to factor
#to allow classification
#all features
noLSDDecade = noLSDDecade[c('Age', 'Gender', 'Education', 'Nscore',
'Escore', 'Oscore', 'Ascore',
                            'Cscore', 'Impulsiveness','SS', 'LSD')]
#just table 18 features
#noLSDDecade = noLSDDecade[c('Age', 'Gender', 'Nscore', 'Escore',
'Oscore', 'Impulsiveness', 'LSD')]

#to get 2:1 for neg:pos class proportions
noLSDDecadeBalanced <- sample_frac(noLSDDecade, 1)

LSDTrainDecade <- rbind(LSDDecade, noLSDDecadeBalanced)

#produces a random forest model named model1.rf
LSDModelDecade.rf <- randomForest(LSD ~., data = LSDTrainDecade,
                                  ntree = ntreeOPT, #how many trees
                                  mtry = mtryOPT,
                                  CUTOFF = CUTOFFOPT, #how many have to
agree for cutoff
                                  importance = TRUE)

#print(paste("optimal number of trees: ", ntreeOpt))
#print(paste("optimal number of variables tried at each node: ", mtryOpt))
```

```
#print(paste("optimal cutoff proportion: ", CUTOFFOPT))
#print(paste("lowest OOB: ",lowestOOB))
print(LSDModelDecade.rf) #display OOB, confusion matrix, and so on for
model1.rf

importance(LSDModelDecade.rf)
varImpPlot(LSDModelDecade.rf)
```

9.5.1 Rebalanced Datasets attempt

```
Call:
 randomForest(formula = Coke ~ ., data = cokeTrainMonth, ntree = ntreeOPT,      mtry = mtryOPT, CUTOFF = CUTOFFOPT, importance
 = TRUE)
               Type of random forest: classification
                     Number of trees: 1000
No. of variables tried at each split: 4

        OOB estimate of  error rate: 30.21%
Confusion matrix:
   1   0 class.error
1 77  82   0.5157233
0 60 251   0.1929260
```

Figure 16: Month-based classification attempt, Cocaine

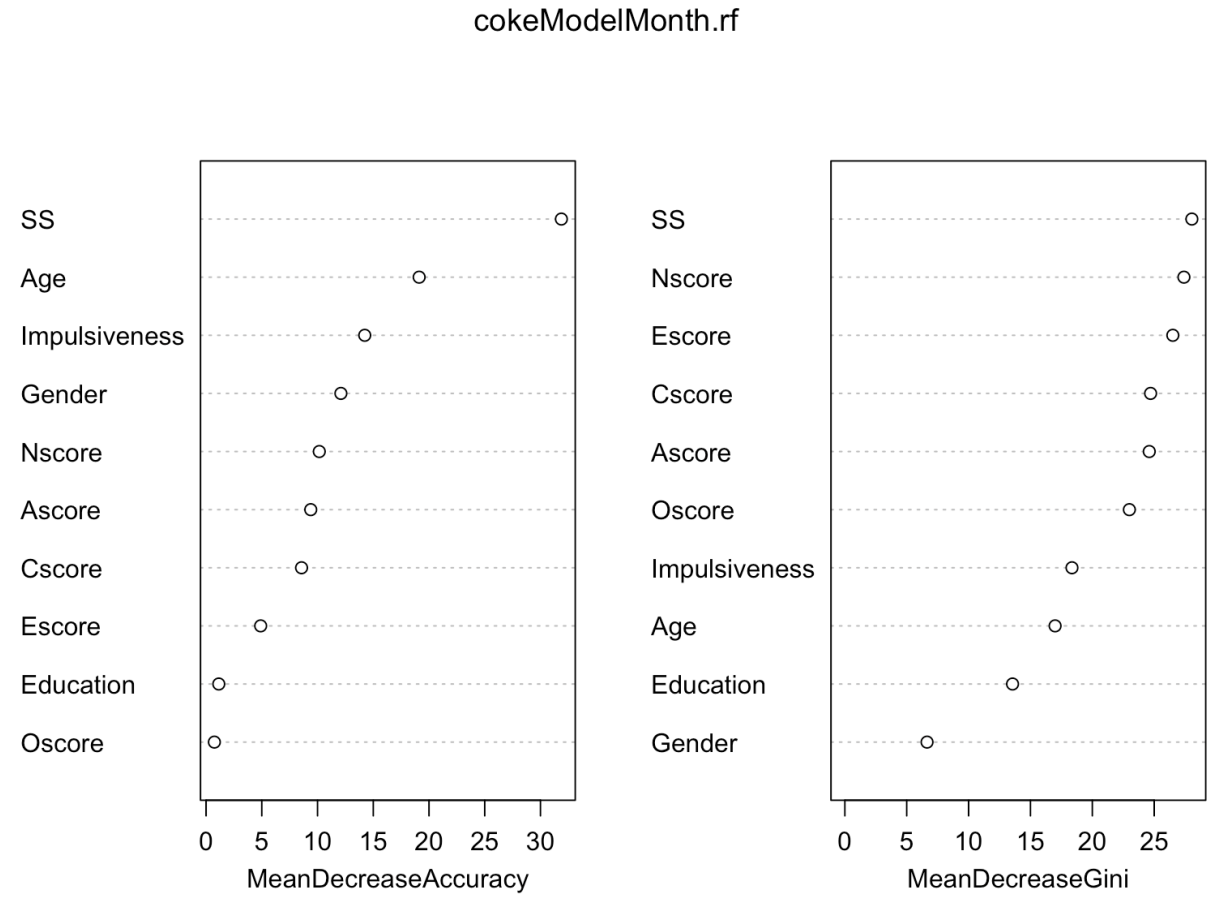


Figure 17: Month-based classification attempt feature rankings, Cocaine

9.5.2 Feature-based Replication Attempt

```
Call:
 randomForest(formula = LSD ~ ., data = LSDTrainMonth, ntree = ntreeOPT,      mtry = mtryOPT, CUTOFF = CUTOFFOPT, importance =
 TRUE)
               Type of random forest: classification
                     Number of trees: 1000
No. of variables tried at each split: 4

        OOB estimate of  error rate: 26.17%
Confusion matrix:
   1   0 class.error
1 96  70   0.4216867
0 59 268   0.1804281
```

Figure 18: Month-based classification attempt, LSD

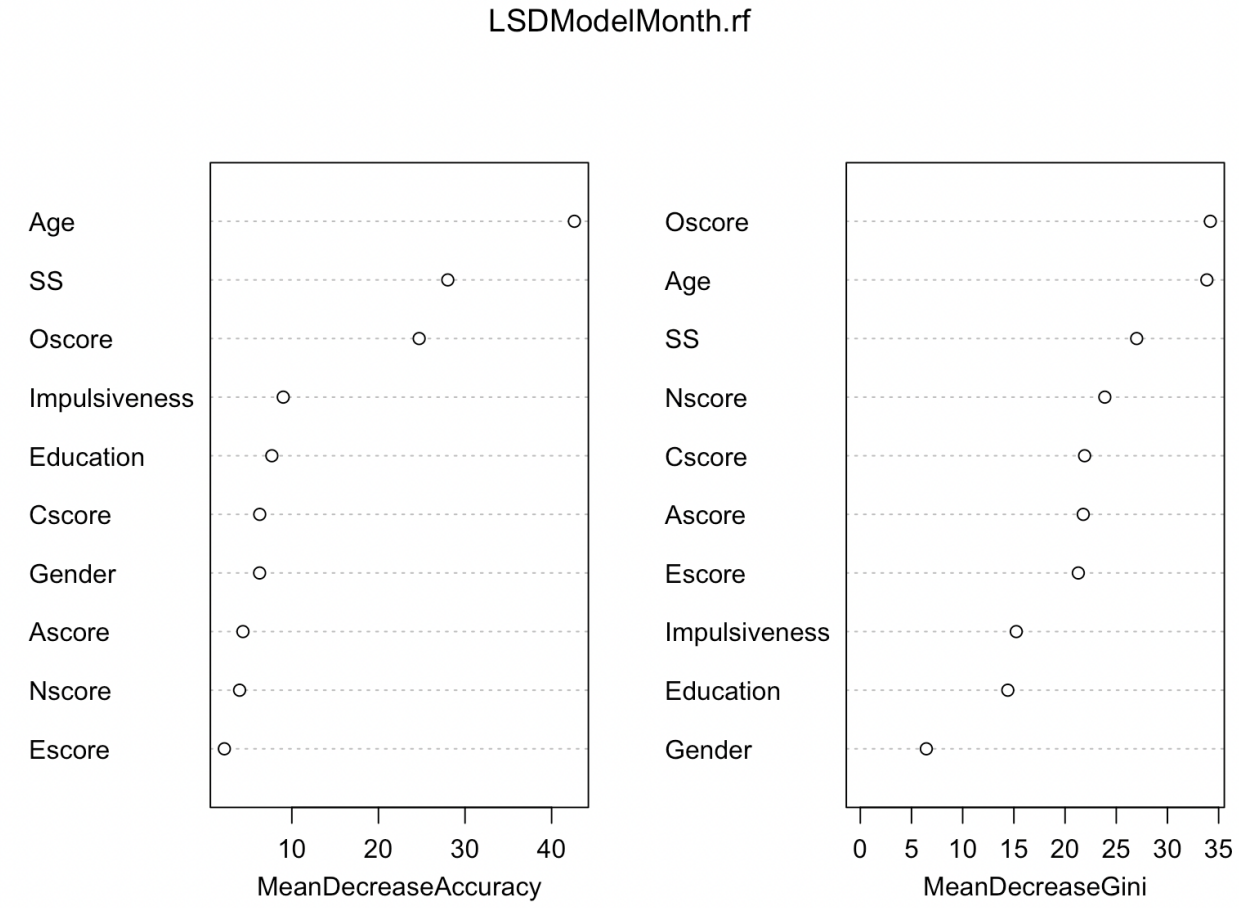


Figure 19: Month-based classification attempt feature rankings, LSD

```
Call:
 randomForest(formula = Mushrooms ~ ., data = MushroomsTrainMonth,      ntree = ntreeOPT, mtry = mtryOPT, CUTOFF = CUTOFFOPT, i
mportance = TRUE)
               Type of random forest: classification
                     Number of trees: 1000
No. of variables tried at each split: 4

        OOB estimate of  error rate: 29.79%
Confusion matrix:
   1   0 class.error
1 77  82   0.5157233
0 58 253   0.1864952
```

Figure 20: Month-based classification attempt, Mushrooms

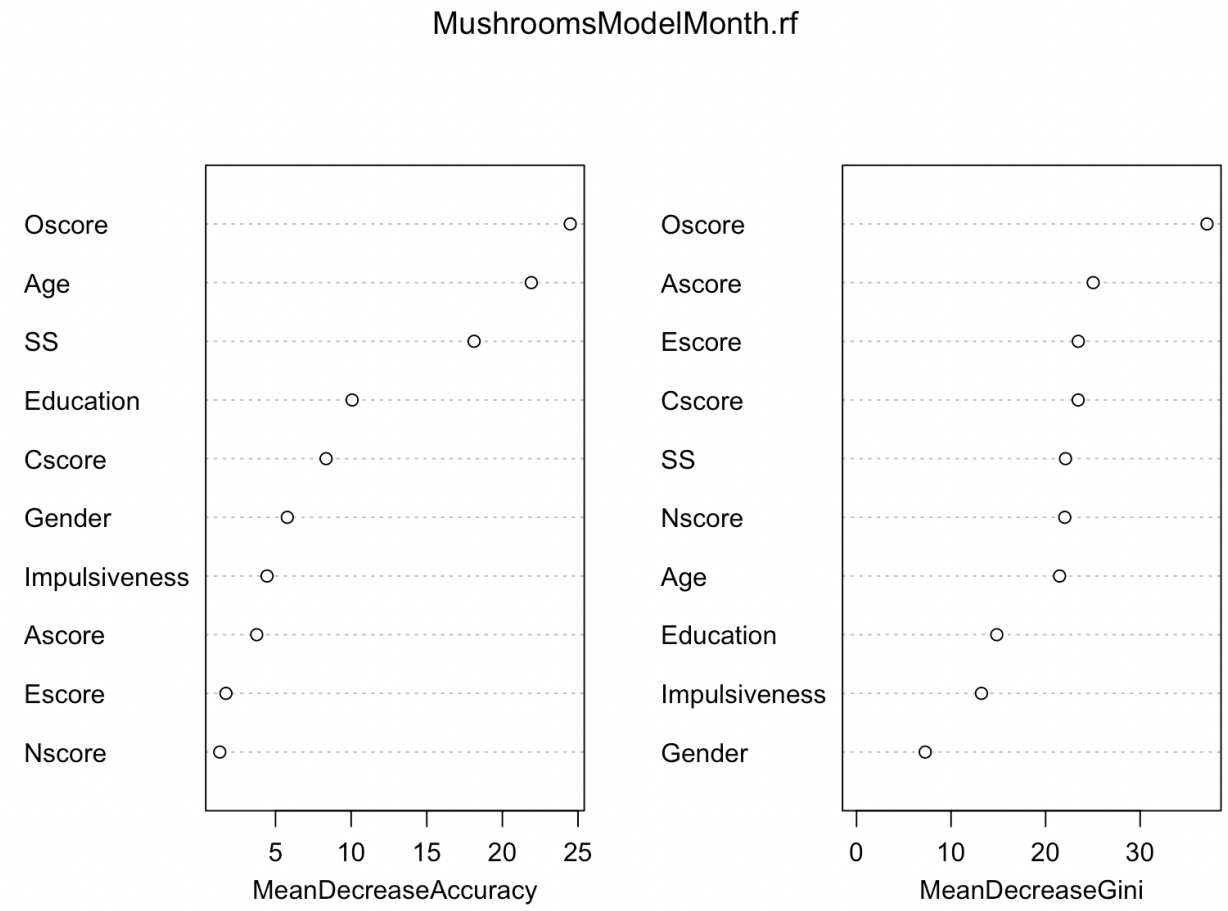


Figure 21: Month-based classification attempt feature rankings, Mushrooms

```
Call:
 randomForest(formula = Meth ~ ., data = MethTrainMonth, ntree = ntreeOPT,      mtry = mtryOPT, CUTOFF = CUTOFFOPT, importance
 = TRUE)
               Type of random forest: classification
                     Number of trees: 1000
No. of variables tried at each split: 4

        OOB estimate of  error rate: 28.57%
Confusion matrix:
   1   0 class.error
1 77  94   0.5497076
0 48 278   0.1472393
```

Figure 22: Month-based classification attempt, Methadone

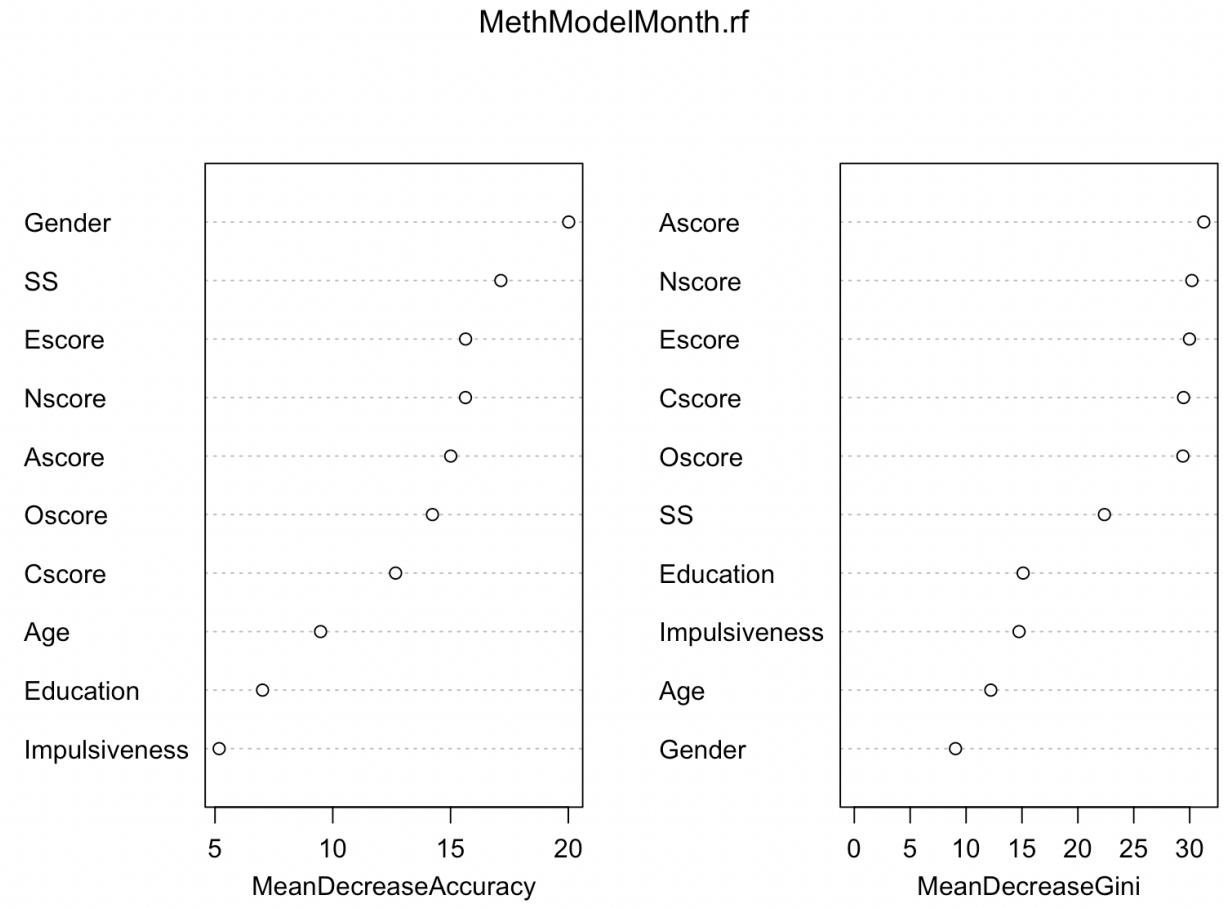


Figure 23: Month-based classification attempt feature rankings, Meth

```
Call:
 randomForest(formula = Crack ~ ., data = CrackTrainMonth, ntree = ntreeOPT,      mtry = mtryOPT, CUTOFF = CUTOFFOPT, importanc
e = TRUE)
               Type of random forest: classification
                     Number of trees: 1000
No. of variables tried at each split: 4

        OOB estimate of  error rate: 26.32%
Confusion matrix:
   1  0 class.error
1 10 10   0.5000000
0  5 32   0.1351351
```

Figure 24: Month-based classification attempt, Crack

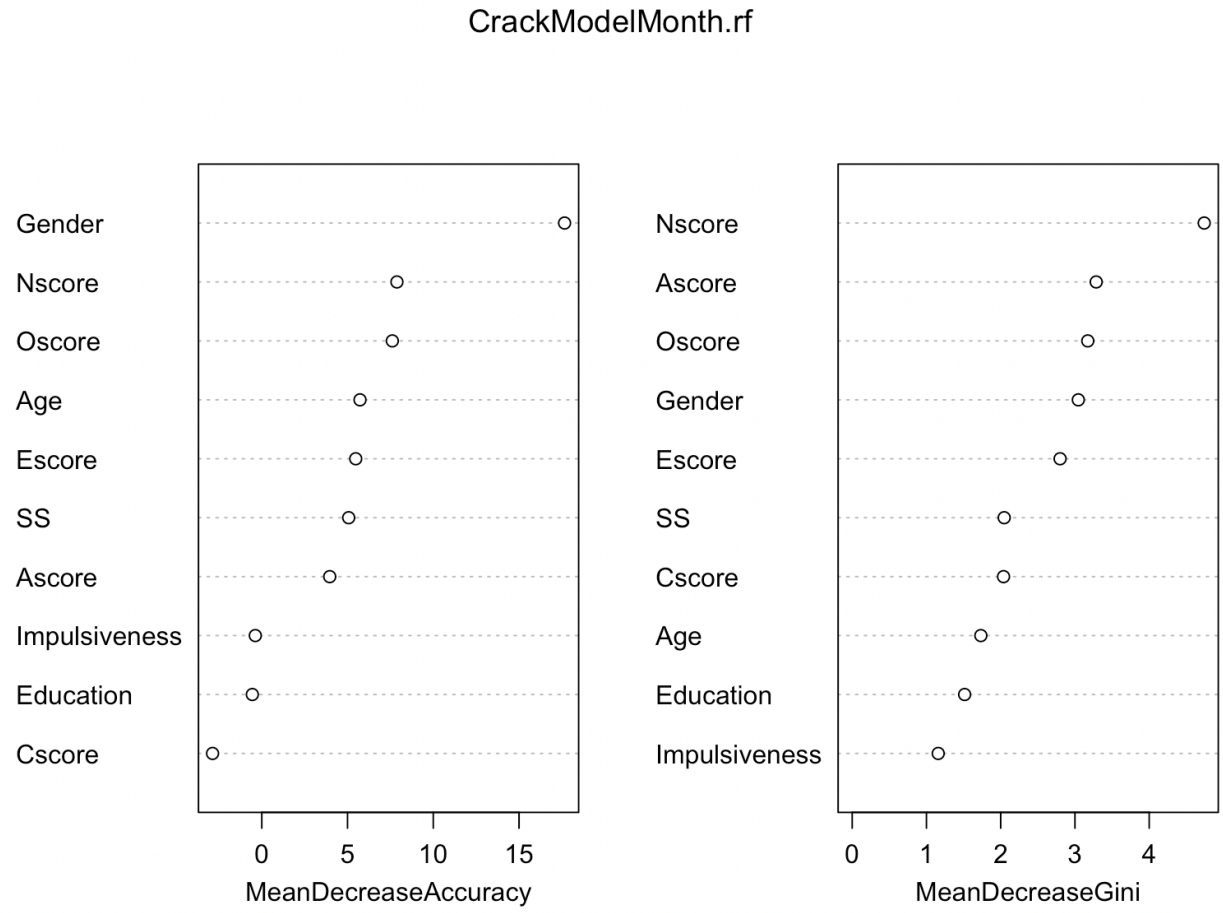


Figure 25: Month-based classification attempt feature rankings, Crack

```
Call:
 randomForest(formula = Heroin ~ ., data = HeroinTrainMonth, ntree = ntreeOPT,      mtry = mtryOPT, CUTOFF = CUTOFFOPT, importa
nce = TRUE)
               Type of random forest: classification
                     Number of trees: 1000
No. of variables tried at each split: 4

        OOB estimate of  error rate: 30.34%
Confusion matrix:
   1  0 class.error
1 28 25   0.4716981
0 19 73   0.2065217
```

Figure 26: Month-based classification attempt, Heroin

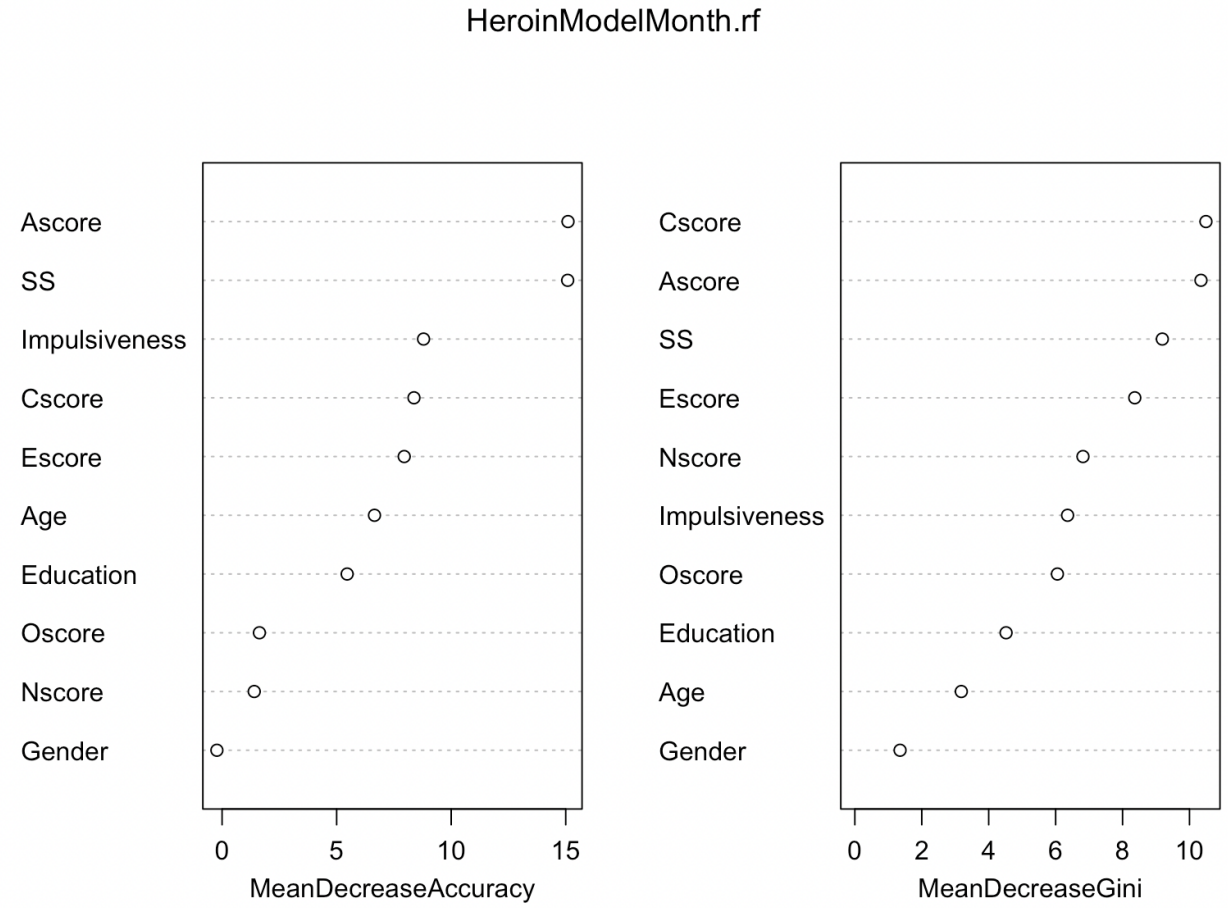


Figure 27: Month-based classification attempt feature rankings, Heroin

```
Call:
 randomForest(formula = Crack ~ ., data = CrackTrainDecade, ntree = ntreeOPT,      mtry = 2, CUTOFF = CUTOF
FOPT, importance = TRUE)
               Type of random forest: classification
                     Number of trees: 1000
No. of variables tried at each split: 2

        OOB estimate of  error rate: 13.63%
Confusion matrix:
   1    0 class.error
1  6  185  0.96858639
0 72 1622  0.04250295
```

Figure 28: Decade-based replication attempt with original features, Crack, mtry = 2

```
Call:
 randomForest(formula = Crack ~ ., data = CrackTrainDecade, ntree = ntreeOPT,      mtry = 1, CUTOFF = CUTOF
FOPT, importance = TRUE)
               Type of random forest: classification
                     Number of trees: 1000
No. of variables tried at each split: 1

        OOB estimate of  error rate: 10.72%
Confusion matrix:
   1    0 class.error
1  1  190 0.994764398
0 12 1682 0.007083825
```

Figure 29: Decade-based replication attempt with original features, Crack, mtry = 1

```
Call:
 randomForest(formula = LSD ~ ., data = LSDTrainDecade, ntree = ntreeOPT,      mtry = 3, CUTOFF = CUTOFFOP
T, importance = TRUE)
               Type of random forest: classification
                     Number of trees: 1000
No. of variables tried at each split: 3

        OOB estimate of  error rate: 24.77%
Confusion matrix:
    1    0 class.error
1 292  265   0.4757630
0 202 1126   0.1521084
```

Figure 30: Decade-based replication attempt with original features, LSD, mtry = 3

```
Call:
 randomForest(formula = LSD ~ ., data = LSDTrainDecade, ntree = ntreeOPT,      mtry = 2, CUTOFF = CUTOFFOP
T, importance = TRUE)
               Type of random forest: classification
                     Number of trees: 1000
No. of variables tried at each split: 2

        OOB estimate of  error rate: 22.86%
Confusion matrix:
    1    0 class.error
1 304  253   0.4542190
0 178 1150   0.1340361
```

Figure 31: Decade-based replication attempt with original features, LSD, mtry = 2