

Deloitte: *Drug* *Case Study*





Table of *contents*



01

What is it?

You can describe the
topic of the section here

02

Our functions

You can describe the
topic of the section here

03

Our numbers

You can describe the
topic of the section here

04

Learn more

You can describe the
topic of the section here



Introduction

Our client at the NIH wants to build a program to address drug use among teenagers/young adults in the US. They are asking you to use existing data to understand factors that lead to drug use and make recommendations for the program, as well as help them understand how and where they should start to roll out these programs.



Overarching Question

What are the top 10 external factors that contribute to hard drug addiction?

When we say hard drugs, we mean any drug “beyond” substances such as alcohol, tobacco, or marijuana.

To what accuracy, could we predict whether someone would use drugs?

Two thin white vertical lines are positioned on the left and right sides of the slide. The left line has a downward-pointing arrowhead at the bottom, and the right line has an upward-pointing arrowhead at the top.

Context



Challenges



Models

PCA? Regularization through
Lasso? XGBoost? Ensemble?



The *data*

2 GB big! 3000 columns



Interpretation

Why did this model work
better than this one?



01

Data Cleaning

Filtering the appropriate data

```
# only want ages 12 - 25  
df = df[(df['CATAGE'] == 1) | (df['CATAGE'] == 2)]  
df = df.set_index('CASEID')
```



Missingness Analysis



Replace columns with majorly
missing values

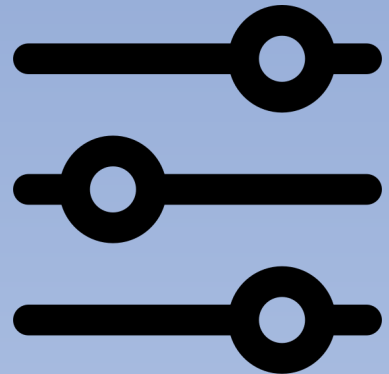
```
df_subset = df.loc[:, df.isna().mean() < 0.95]  
df_subset
```

Filtering Relevant Columns

- Stimulants, Crack, Cocaine, Heroin, Hallucinogens, Pain Relievers, Tranquilizers, Sedative, Marijuana



Drug Abuse?



02



Exploratory Data Analysis

You can enter a subtitle here if you need it

Storytelling reports



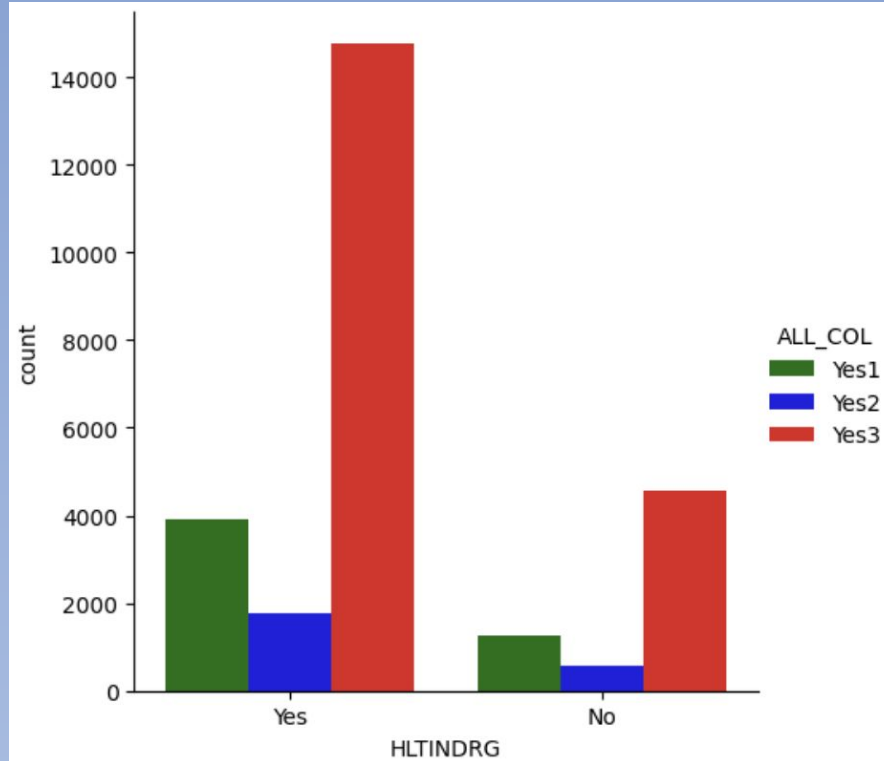
About the Data



Data *Visualization*


Data storytelling reporting

Temp
Graphic of
EDA, will
explain
variables





How we chose variables



```
In [20]: importance_df.sort_values(by = "importance", ascending = False)
#this only caught 2 features as non zero
# HLTINALC HLTNMNT which is only marijuana (was hoping for more)
```

Out[20]:

	feature	importance
1731	HLTINALC	0.060461
1732	HLTINMNT	0.036584
0	CASEID	0.0
1229	YETCGJOB	0.0
1227	YESCHIMP	-0.0
...
606	RKHERREG	0.0
605	RKTRYHER	0.0
604	RKLSDREG	0.0
603	RKTRYLSD	0.0
1829	VEREP	-0.0

1830 rows x 2 columns

ONLY 2
VARIABLES
ARE
IMPORTANT?



In [23]: `pca.pca(X ,X_train)`

There are 837 components that explain the variance within the dataset

STMMON: 0.015704350903006103

CPNSTMMN: 0.015669309736575845

IICRKRC: 0.015087308435441616

II2CRKRC: 0.014955312294304399

ABUSEHER: 0.014903875549945574

ILLPSAVE: 0.0148607790313776

DRGTXER: 0.014802996842752176

IIEC SRC: 0.014673944710285516

SPILANAL: 0.014645522269818474

Stimulants

Crack



How we chose variables

- PCA? Not good for categorical
- Lasso?

Solution: XGBoost feature importance attribute

Why XGBoost?

- Built in feature importance attribute
 - Easy to access and evaluate
 - Pipeline Objects allow for easy implementation
- 

03

Different Models

Talk About Feature Selection,
Lasso, Ridge, and Logistic, PCA



```
# Create a Logistic Regression model with L2 regularization
lr = LogisticRegression(penalty='l2')
train_and_evaluate_model(lr, X, y, "logistic_regression_model", categorical_vars)
```

✓ 54.2s

/Users/palvins/.pyenv/versions/3.9.4/lib/python3.9/site-packages/sklearn/linear_model/_l

lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

/Users/palvins/.pyenv/versions/3.9.4/lib/python3.9/site-packages/sklearn/metrics/_classi

Precision is ill-defined and being set to 0.0 due to no predicted samples. Use `zero_div

Confusion Matrix:

```
[[5444   0]
 [1732   0]]
```

Accuracy: 0.7586399108138239

Precision: 0.0

Recall: 0.0

F1 Score: 0.0

Logistic Regression & Ridge Regularization

```
# Create a Random Forest classifier
rf_clf = RandomForestClassifier(n_estimators=100)
result = train_and_evaluate_model(rf_clf, X, y, "random_forest_model", categorical_variables)
result
```

✓ 2m 38.2s

Output exceeds the [size limit](#). Open the full output data [in a text editor](#)

Confusion Matrix:

```
[[5439   5]
 [1685  47]]
```

Accuracy: 0.7644927536231884

Precision: 0.9038461538461539

Recall: 0.027136258660508082

F1 Score: 0.052690582959641255

Cross-validation scores:

```
[0.76658305 0.76755853 0.77912486 0.76989547 0.768223 ...]
```

Mean cross-validation score:

0.7702769821200363

Feature: HLTINALC, Importance: 0.022779275697964497

Feature: IRPRVHLT, Importance: 0.0029071884796050412

Feature: HLTINMNT, Importance: 0.016500114963150243

Feature: AMHINP2, Importance: 0.00014220000642720652

Feature: AMHRX2, Importance: 0.00024028750979454192

Feature: AMHTXRC3, Importance: 0.0002837014069168502

Feature: AMHSVTYP, Importance: 0.00028337778147641236

Feature: PRVHLTIN, Importance: 0.002640207211520947

	Feature	Importance
0	HLTINALC	0.022779
2	HLTINMNT	0.016500
244	ANALWT_C	0.004471
500	BMI2	0.004467
435	WTPOUND2	0.003831
...
1300	TUINAL2	0.000000
1296	CHHYD2	0.000000
1377	INNEVER	0.000000
1294	PRELUDN2	0.000000
1291	ROHYPNL2	0.000000

Random Forest

Boosted Decision Trees (using XGBoost)

The boosting technique is powerful, but it can lead to overfitting if not used with care. It's impor

```
# create XGBClassifier
xg_clf = xgb.XGBClassifier(n_estimators=100, learning_rate=0.1, max_depth=5)
result = train_and_evaluate_model(xg_clf, X, y, "xgboost_model", categorical_vars)
result
```

✓ 4m 45.5s

Output exceeds the [size limit](#). Open the full output data [in a text editor](#)

Confusion Matrix:

[[5369 75]

[1360 372]]

Accuracy: 0.8000278706800446

Precision: 0.8322147651006712

Recall: 0.21478060046189376

F1 Score: 0.3414410279944929

Cross-validation scores:

[0.80351171 0.80504459 0.80490524 0.80334495 0.79902439]

Mean cross-validation score:

0.8031661752881265

Feature: HLTINALC, Importance: 0.04634871333837509

Feature: IRPRVHLT, Importance: 0.01709429733455181

Feature: HLTINMNT, Importance: 0.019346199929714203

Feature: AMHINP2, Importance: 0.005094137508422136

Feature: AMHRX2, Importance: 0.006371975410729647

Feature: AMHTXRC3, Importance: 0.00412711501121521

Feature: AMHSVTYP, Importance: 0.007858609780669212

Feature: PRVHLTIN, Importance: 0.005835204850882292

Feature: IRTNRNC, Importance: 0.0027723219245672226

Feature: IINHAGE, Importance: 0.002947981469333172

XGBoost

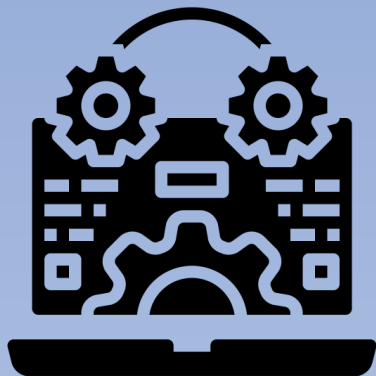
	Feature	Importance
1746	HLTINALC	0.028191
1759	IRPRVHLT	0.015407
1747	HLTINMNT	0.013420
1164	AMHINP2	0.006535
1166	AMHRX2	0.005058
...
1681	UADOTHM	0.000228
1581	YOWRLSIN	0.000189
675	ALCCUTDN	0.000159
872	ANLFR TK2	0.000101
357	IEMFLAG	0.000021

04 Final Model



Model Optimization

- GridSearchCV
- 60 Total Combinations
- ~3-4 minutes per model



```
parameters = {  
    'n_estimators': range(50, 250, 50),  
    'learning_rate': [0.01, 0.1, 0.3],  
    'max_depth': range(1, 11, 2)  
}  
clf = GridSearchCV(xg_clf, param_grid=parameters)  
grid_result = clf.fit(X_train, y_train)  
grid_result.best_estimator_
```


05

Most Important Factors

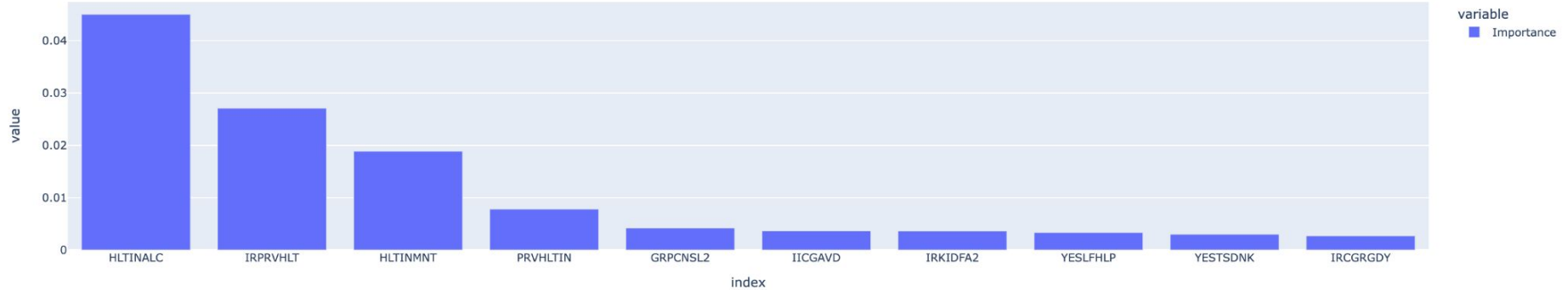


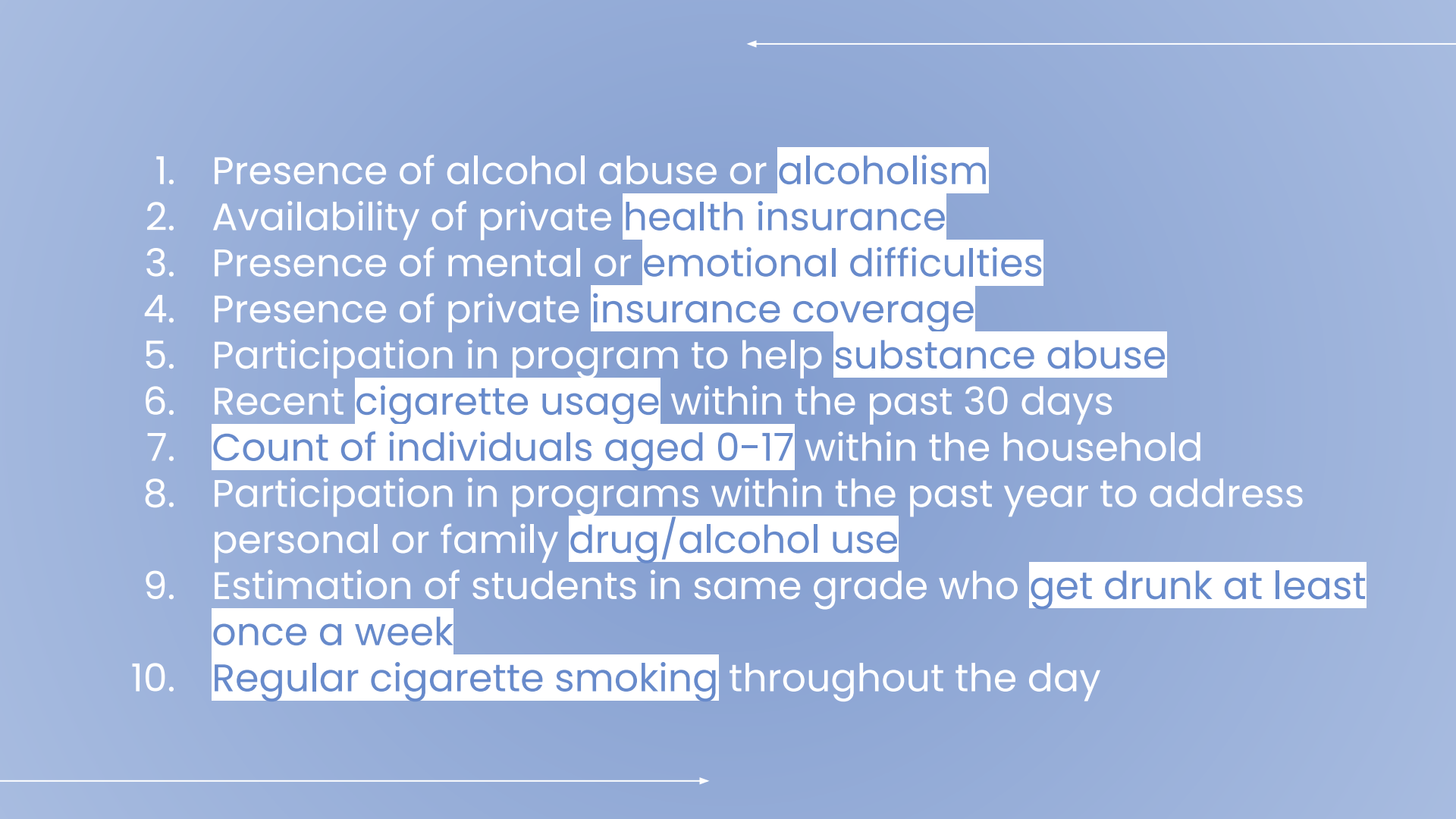
Top 10 Features and Their Importance

	Feature	Importance
1.	HLTINALC	0.045076
2.	IRPRVHLT	0.027089
3.	HLTINMNT	0.018880
4.	PRVHLTIN	0.007793
5.	GRPCNSL2	0.004189
6.	IICGAVD	0.003621
7.	IRKIDFA2	0.003589
8.	YESLFHLP	0.003303
9.	YESTSDNK	0.002986
10.	IRCGRGDY	0.002688

Temp Graphic of Results

Top Ten Important Features Using XGBoost Model



- 
1. Presence of alcohol abuse or alcoholism
 2. Availability of private health insurance
 3. Presence of mental or emotional difficulties
 4. Presence of private insurance coverage
 5. Participation in program to help substance abuse
 6. Recent cigarette usage within the past 30 days
 7. Count of individuals aged 0-17 within the household
 8. Participation in programs within the past year to address personal or family drug/alcohol use
 9. Estimation of students in same grade who get drunk at least once a week
 10. Regular cigarette smoking throughout the day



Major Categories



1. **Substance Abuse and Addiction**
 - a. Alcohol
 - b. Nicotine
2. **Mental Health**
3. **Healthcare**
4. **Support and Intervention**



Evaluation:

Precision, Recall, F1, Accuracy -> Which Model is the 'Best'

- Accuracies were decent, but that's not the only metric we should focus on.
- F1 is a more reliable metric since our data was imbalance

Model trained on imputed data: F1 score was .34 (not good)

At the last minute, we decided to try a model trained on non-imputed data

F1 score >.95 (overfitting?)

```
In [*]: # create XGBClassifier
xg_clf = xgb.XGBClassifier(n_estimators=100, learning_rate=0.1, max_depth=5)
result = train_and_evaluate_model(xg_clf, X, y, "xgboost_model", categorical_vars)
result
```

Confusion Matrix:

```
[[1614    9   69]
 [    9  485   27]
 [   51   18 4894]]
```

Accuracy: 0.9744983277591973

We used an imputation parameter within XGBoost

- The list of most important features changed significantly
- Most likely due to a large number of NA values (3100 features since we weren't imputing)

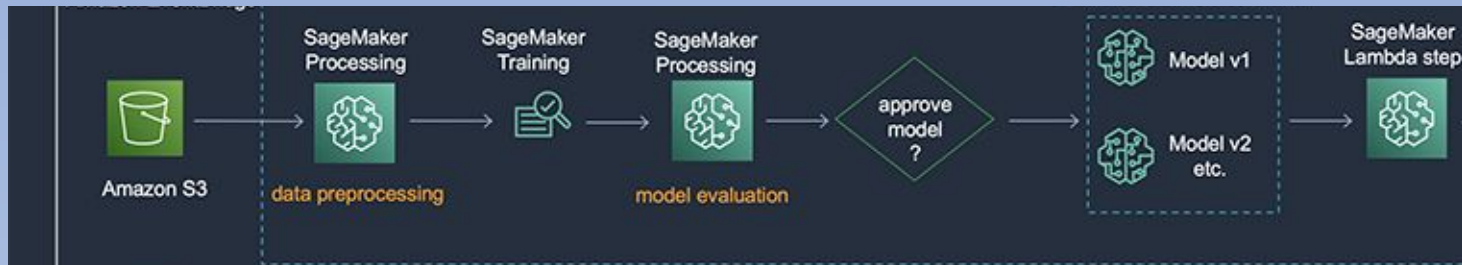
What does this mean?

- We need to go back and analyze the columns
- Try different algorithms without imputing

Once we improve model's performance (F1 score, Accuracy), where do we go from here?

- Deploy it on the cloud using a service like Sagemaker
- Create a pipeline to feed new data in using AWS Lambda

Repeat the cycle as time goes on



Conclusion



Las Vegas



New Orleans

