Module 3 assignment-resubmission

October 31, 2024

1 Module 3 - Algorithm auditing: Accuracy, Fairness and Interpretability

1.0.1 Assignment overview

In this assignment, you will be asked to evaluate a set of trained classifiers for accuracy, fairness and transparency. The classifiers have been trained on the NIJ Recidivism Challenge Dataset to predict whether or not an individual will be arrested for a new crime within 3 years after being released on parole.

The assignment is modeled after "Accuracy, Fairness, and Interpretability of Machine Learning Criminal Recidivism Models, by Eric Ingram, Furkan Gursoy, Ioannis A. Kakadiaris (https://arxiv.org/abs/2209.14237).

For this assignment, it is possible to work in **groups of up to 2 students**. Read the instructions carefully, as they may assign tasks to specific students.

1.0.2 Group members

Leave blanks if group has less than 2 members: - Student 1: Yanxin Liang 50798412 - Student 2: Yelia Ye 89657605

1.0.3 Learning Goals:

After completing this week's lecture and tutorial work, you will be able to: 1. Describe different fairness metrics, such as statistical parity, equal opportunity and equal accuracy 2. Discuss fairness and fairness metrics from the perspective of multiple stakeholders 3. Define objective functions based on fairness metrics
4. Evaluate a model's transparency using strategies such as global surrogate models, permutation feature importance, and Shapley Additive Explanations (SHAP) 5. Evaluate common machine learning models based on their accuracy, fairness and interpretability 6. Describe how metrics such as accuracy and fairness need to be balanced for a trained model to have acceptable accuracy and low bias

1.1 Import Libraries:

[ ]:
```
# Here are some libraries you may need for this exercise, for your convenience

import matplotlib.pyplot as plt
import numpy as np
```

```
import pandas as pd
#import seaborn as sns
import xgboost as xgb
from sklearn.compose import ColumnTransformer, make_column_transformer
from sklearn.dummy import DummyClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.impute import SimpleImputer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import (
    classification_report,
    confusion_matrix,
    #plot_confusion_matrix,
    f1_score,
    make_scorer,
    ConfusionMatrixDisplay,
    accuracy_score, precision_score, recall_score, roc_auc_score,␣
 ↪confusion_matrix
)
from sklearn.model_selection import (
    GridSearchCV,
    RandomizedSearchCV,
    cross_val_score,
    cross_validate,
    train_test_split,
)
from sklearn.pipeline import Pipeline, make_pipeline
from sklearn.preprocessing import OneHotEncoder, OrdinalEncoder, StandardScaler
from sklearn.tree import export_text


import joblib
from sklearn import tree
from sklearn.inspection import permutation_importance
import eli5

import warnings
warnings.filterwarnings("ignore")
```

2 Part 1: Getting started:

Before starting this assignment, we ask you to read the paper it has been modeled after, to get an idea of the problem we are working on: https://arxiv.org/abs/2209.14237

You can also review the original dataset source here. The website includes a lot of information on the dataset and a detailed description of each of its columns (look for Appendix 2: Codebook).

Now that you have familiarized with the problem, you know that the goal is predicting the binary variable `Recidivism_Within_3years`, which indicates whether or not the person has committed a

new felony or misdemeanour within 3 years from the beginning of parole supervision.

The National Institute of Justice's (NIJ) obviously would want to deploy a highly accurate predictive model, to make sure that only deserving people get released on parole. Unfortunately, the existence of bias in the training set (typically historical or representation bias) makes it very likely to end up with an unfair classifier, that is, a classifier that produces different results for different protected classes of population.

Your job is to evaluate 5 classifiers, pre-trained and provided to you. This is called **algorithm auditing:** you are not the designer of the model, but you are in charge of evaluating its performance. Algorithm auditing can focus on various metrics and populations of interest, but in this case we will focus on evaluating **accuracy, fairness and transparency** of each algorithm.

To begin, load the datasets and classsifiers by running the cells below:

[ ]:
```
# Note: these training and test sets do not correspond to the ones on the NIJ's␣
 ↪website,
# they are our own partition

train_df = pd.read_csv("training_set.csv")
test_df = pd.read_csv("testing_set.csv")
```

[ ]:
```
# Creating training and test sets and separating features and target
X_train, y_train = (
    train_df.drop(columns=["Recidivism_Within_3years"]),
    train_df["Recidivism_Within_3years"],
)
X_test, y_test = (
    test_df.drop(columns=["Recidivism_Within_3years"]),
    test_df["Recidivism_Within_3years"],
)
```

[ ]:
```
# Loading classifiers
logreg_model  = joblib.load("models_for_A3/NIJ_logreg.joblib")
rf_model      = joblib.load("models_for_A3/NIJ_rf.joblib")
tree_model    = joblib.load("models_for_A3/NIJ_tree.joblib")
xgboost_model = joblib.load("models_for_A3/NIJ_xgboost.joblib")
```

3 Part 2: Classifiers' Accuracy (and other performance metrics):

First, we will evaluate each classifier's accuracy, together with other performance metrics that help us understanding how reliable the classifier's answers are. In addition to accuracy, we will use, **precision, recall, F1 score, and Area Under the Curve (AUC).**

3.0.1 Question 1

can you provide definition and formula for accuracy, precision, recall and F1 score?

It may help you use this table for reference:

Here, we are giving you the definition of AUC, as a reminder and example (note that the other metrics will need the formula):

AUC: AUC stands for Area Under the ROC curve. The ROC (receiver operating characteristic) curve is a plot of the recall and false positive rate of a classifier for different classification thresholds (see here for more details). AUC values go between 0 and 1. Higher values are more desirable as they indicate that the classifier is good at avoiding both false positives and false negatives. A value of 0.5 for a binary classification indicates that the classifier is no better at predicting the outcome than random guessing.

Accuracy: accuracy is the proportion of the correct predictions:

$$accuracy = \frac{correct\ predictions}{total\ examples} = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision: precision is the proportion of true positive predictions out of all predicted positive cases.:

$$precision = \frac{correct\ positive\ predictions}{all\ predicted\ positive\ cases} = \frac{TP}{TP + FP}$$

Recall: recall is the proportion of the correct positive prediction in all actual cases. It is also called sensitivity, coverage, true positive rate (TPR):

$$recall = \frac{correct\ positive\ predictions}{all\ actual\ cases} = \frac{TP}{TP + FN}$$

F1 score: F1-score is a harmonic mean of precision and recall. F1-score combines precision and recall to give one score:

$$f1 = 2 \times \frac{precision \times recall}{precision + recall}$$

3.0.2 Question 2

For every classifier given, calculate and report accuracy, precision, recall, F1 score, and AUC on both training and test set. **For ease of visualization, summarize these results in one or two tables below this question.**

Hints: - Scikit-learn provides a lot of useful built-in functions to compute performance metrics. You can find them all in the package `sklearn.metrics`, under Classification Metrics. - Some classifiers may take longer than others to make their predictions, so you may have to wait a few minutes for a cell to run. More than that, however, likely means something is wrong and needs to be fixed before continuing.

3.0.3 LogReg Model:

[ ]:
```
def calculate_metrics(y_true, y_pred, y_proba):
    metrics = {
        "Accuracy": accuracy_score(y_true, y_pred),
        "Precision": precision_score(y_true, y_pred, average='binary'),
        "Recall": recall_score(y_true, y_pred, average='binary'),
        "F1 Score": f1_score(y_true, y_pred, average='binary'),
```

```
        "AUC": roc_auc_score(y_true, y_proba)
    }
    return metrics
```

[ ]:
```
train = calculate_metrics(y_train, logreg_model.predict(X_train),  logreg_model.
 ↪predict_proba(X_train)[::,1])
test = calculate_metrics(y_test, logreg_model.predict(X_test),  logreg_model.
 ↪predict_proba(X_test)[::,1])

def metric_df(traindf,testdf):
    d1 = pd.DataFrame.from_dict(traindf, orient='index', columns=['Train'])
    d2 = pd.DataFrame.from_dict(testdf, orient='index', columns=['Test'])
    return pd.concat([d1, d2], axis=1)

metric_df(train,test)
```

[ ]:

	Train	Test
Accuracy	0.714610	0.701974
Precision	0.776119	0.749397
Recall	0.716321	0.709685
F1 Score	0.745022	0.729000
AUC	0.785838	0.773387

3.0.4 Random Forest Model:

[ ]:
```
train_rf = calculate_metrics(y_train, rf_model.predict(X_train),  rf_model.
 ↪predict_proba(X_train)[::,1])
test_rf = calculate_metrics(y_test, rf_model.predict(X_test),  rf_model.
 ↪predict_proba(X_test)[::,1])
metric_df(train_rf,test_rf)
```

[ ]:

	Train	Test
Accuracy	0.996627	0.719262
Precision	0.999332	0.721262
Recall	0.994870	0.819781
F1 Score	0.997096	0.767372
AUC	0.999925	0.774727

3.0.5 Decision Tree Model:

[ ]:
```
train_tree = calculate_metrics(y_train, tree_model.predict(X_train), ␣
 ↪tree_model.predict_proba(X_train)[::,1])
test_tree= calculate_metrics(y_test, tree_model.predict(X_test),  tree_model.
 ↪predict_proba(X_test)[::,1])
metric_df(train_tree,test_tree)
```

[ ]:

	Train	Test
Accuracy	0.738609	0.697845
Precision	0.761806	0.713596
Recall	0.801539	0.776839
F1 Score	0.781168	0.743876
AUC	0.810035	0.753538

3.0.6 XGBoost Model:

[ ]:
```
train_xgboost = calculate_metrics(y_train, xgboost_model.predict(X_train), ␣
↪xgboost_model.predict_proba(X_train)[::,1])
test_xgboost = calculate_metrics(y_test, xgboost_model.predict(X_test), ␣
↪xgboost_model.predict_proba(X_test)[::,1])
metric_df(train_xgboost,test_xgboost)
```

[ ]:

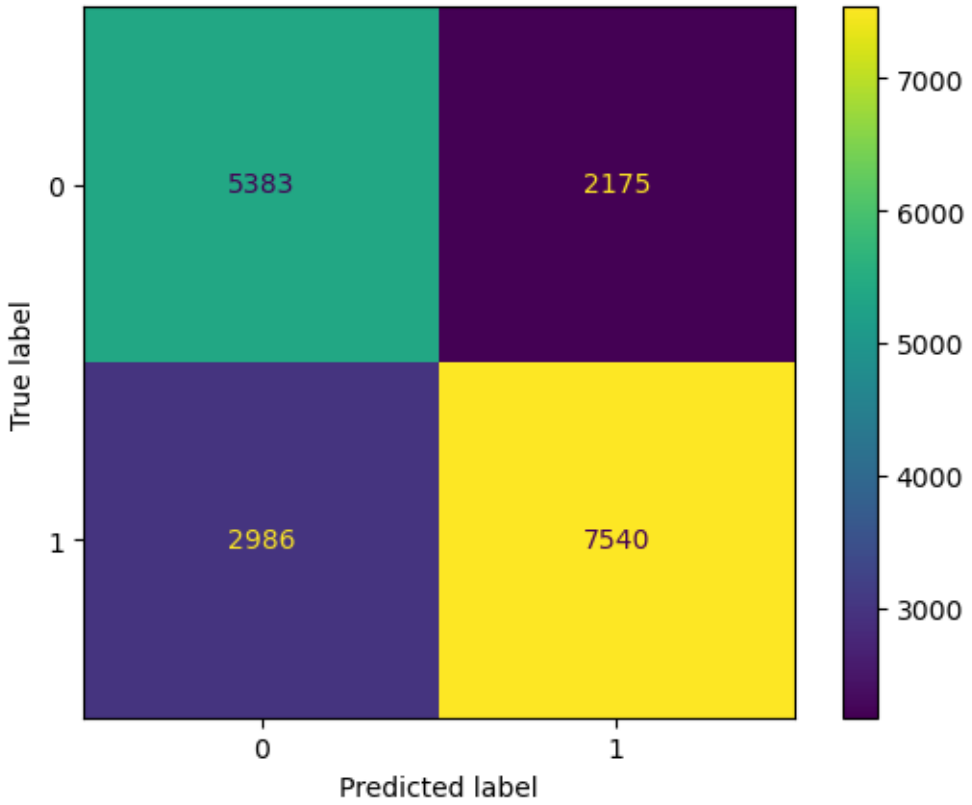
	Train	Test
Accuracy	0.873590	0.735905
Precision	0.867201	0.737228
Recall	0.924378	0.827318
F1 Score	0.894877	0.779679
AUC	0.948941	0.809694

3.0.7 Question 3

For every classifier given, plot the confusion matrices on training and test set. Here is another function you will find helpful for this task: `confusion_matrix`.

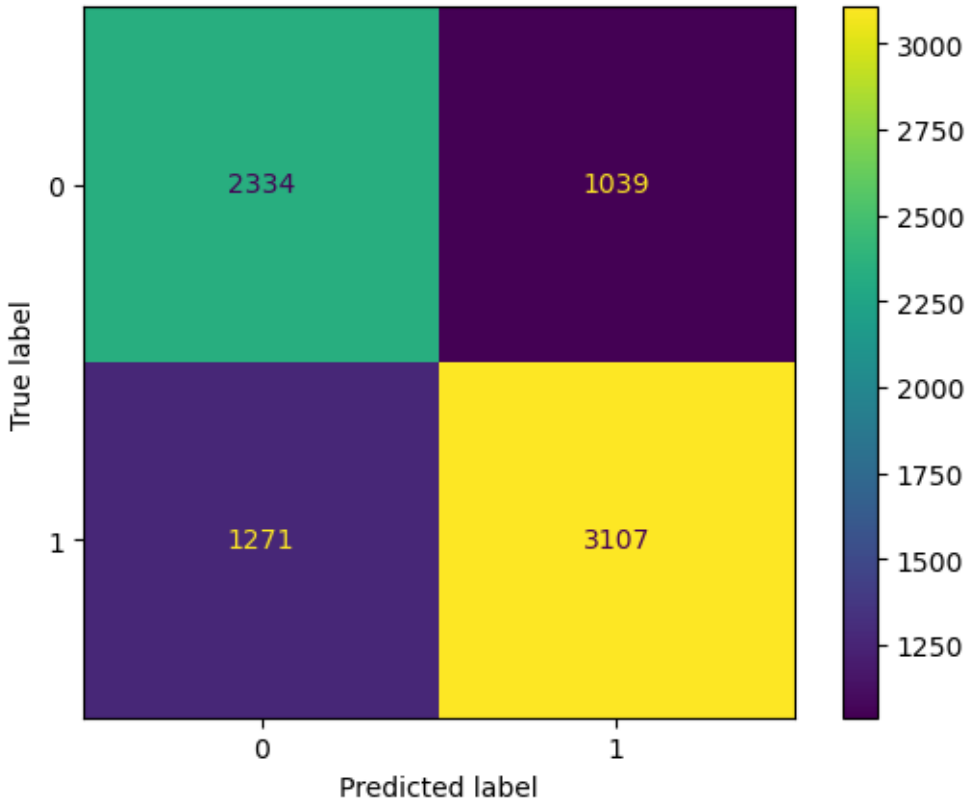
[ ]:
```
#confusion matrices of logreg_model for train set
cm_logreg_train=confusion_matrix(y_train, logreg_model.predict(X_train))
cm_logreg_test=confusion_matrix(y_test, logreg_model.predict(X_test))

ConfusionMatrixDisplay(cm_logreg_train).plot()
```

[ ]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x29c1018d480>
```



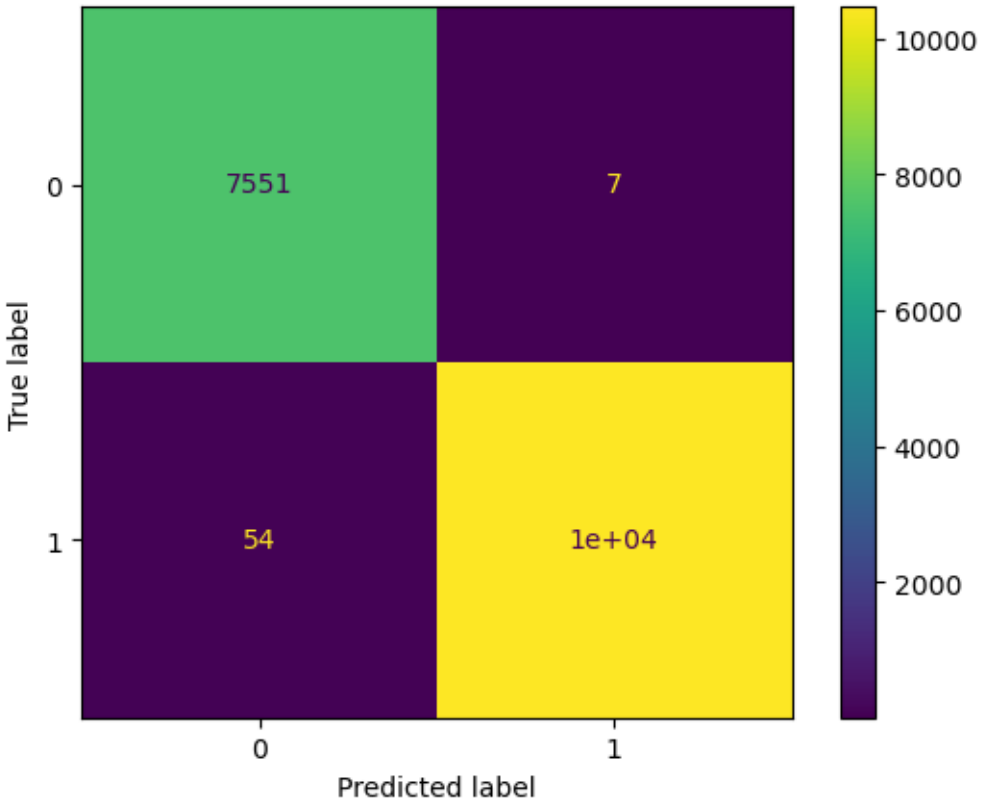
[ ]:
```
#confusion matrices of logreg_model for test set
ConfusionMatrixDisplay(cm_logreg_test).plot()
```

[ ]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x29c161f5ea0>
```



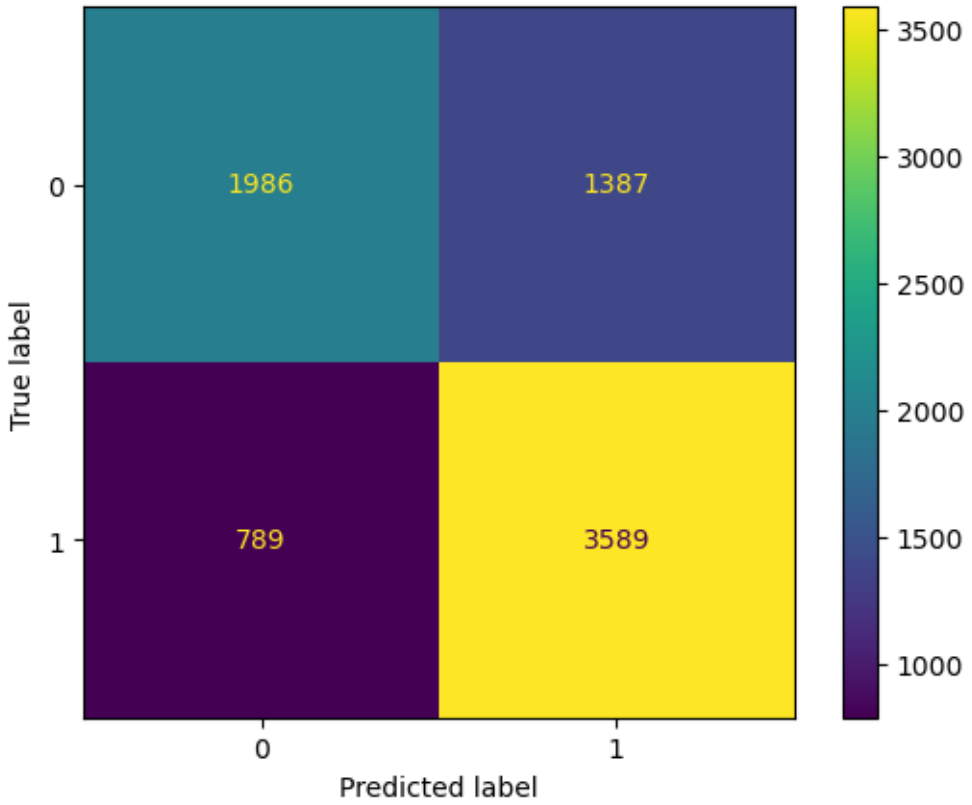
[ ]:
```
#confusion matrices of rf_model for train set
cm_rf_train=confusion_matrix(y_train, rf_model.predict(X_train))
ConfusionMatrixDisplay(cm_rf_train).plot()
```

[ ]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x29c161f7160>
```



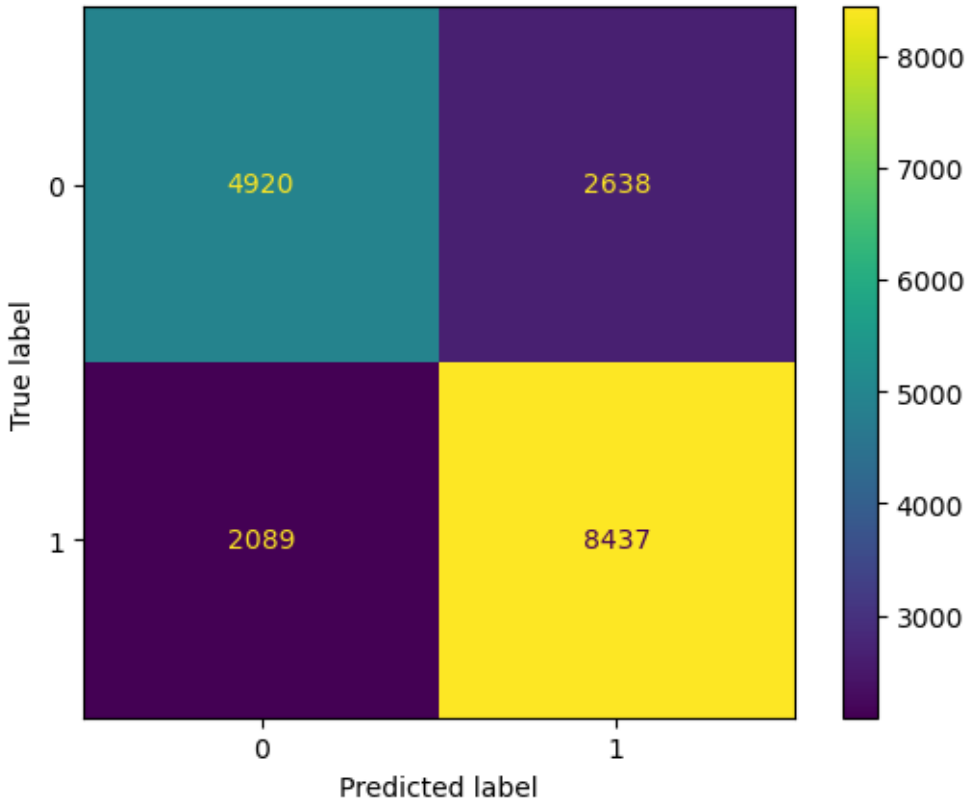
[ ]:
```
#confusion matrices of rf_model for test set
cm_rf_test=confusion_matrix(y_test, rf_model.predict(X_test))
ConfusionMatrixDisplay(cm_rf_test).plot()
```

[ ]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x29c162be410>
```



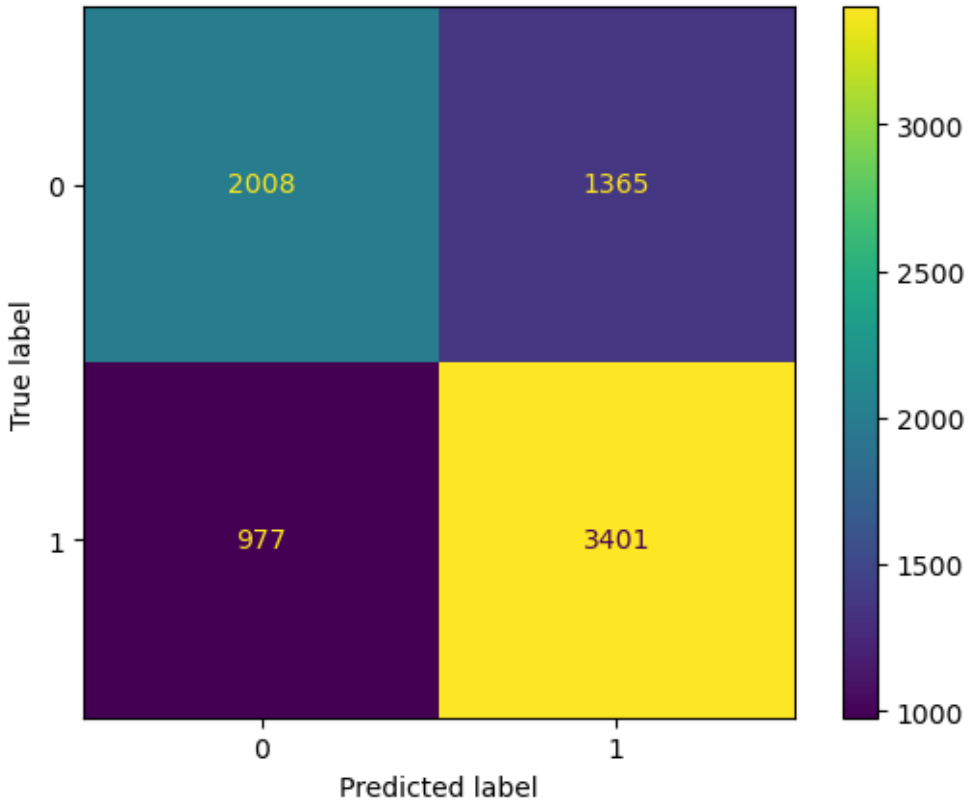
[ ]:
```
#confusion matrices of tree_model for train set
cm_tree_train=confusion_matrix(y_train, tree_model.predict(X_train))
ConfusionMatrixDisplay(cm_tree_train).plot()
```

[ ]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x29c1666ded0>
```



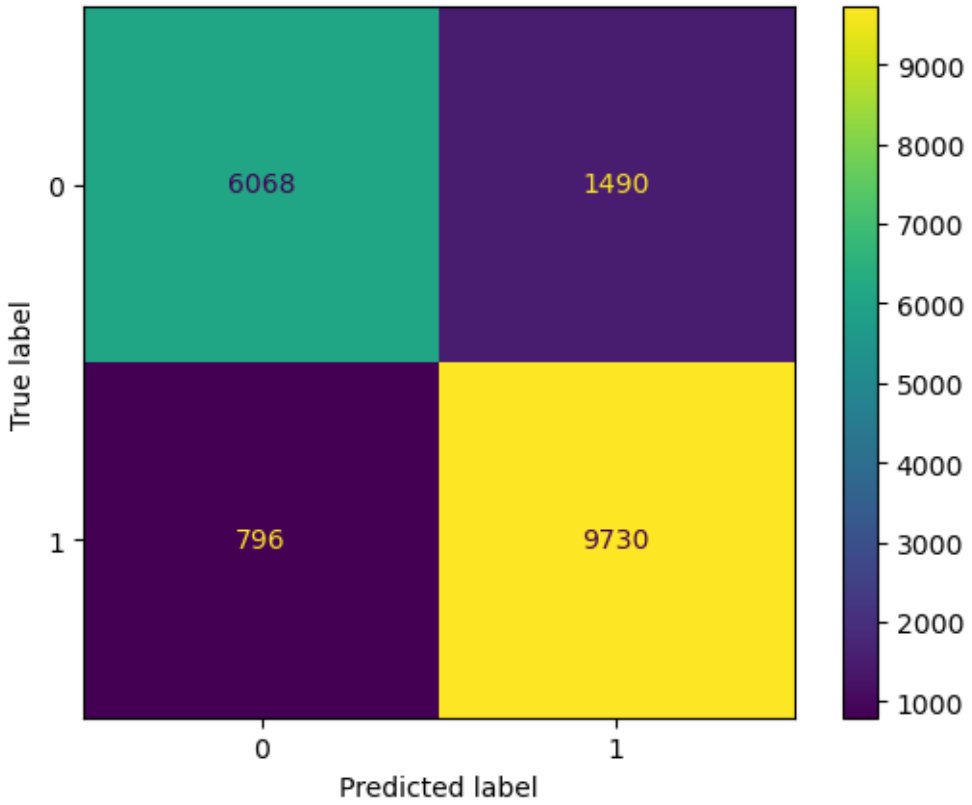
[ ]:
```
#confusion matrices of tree_model for test set
cm_tree_test=confusion_matrix(y_test, tree_model.predict(X_test))
ConfusionMatrixDisplay(cm_tree_test).plot()
```

[ ]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x29c16716530>
```



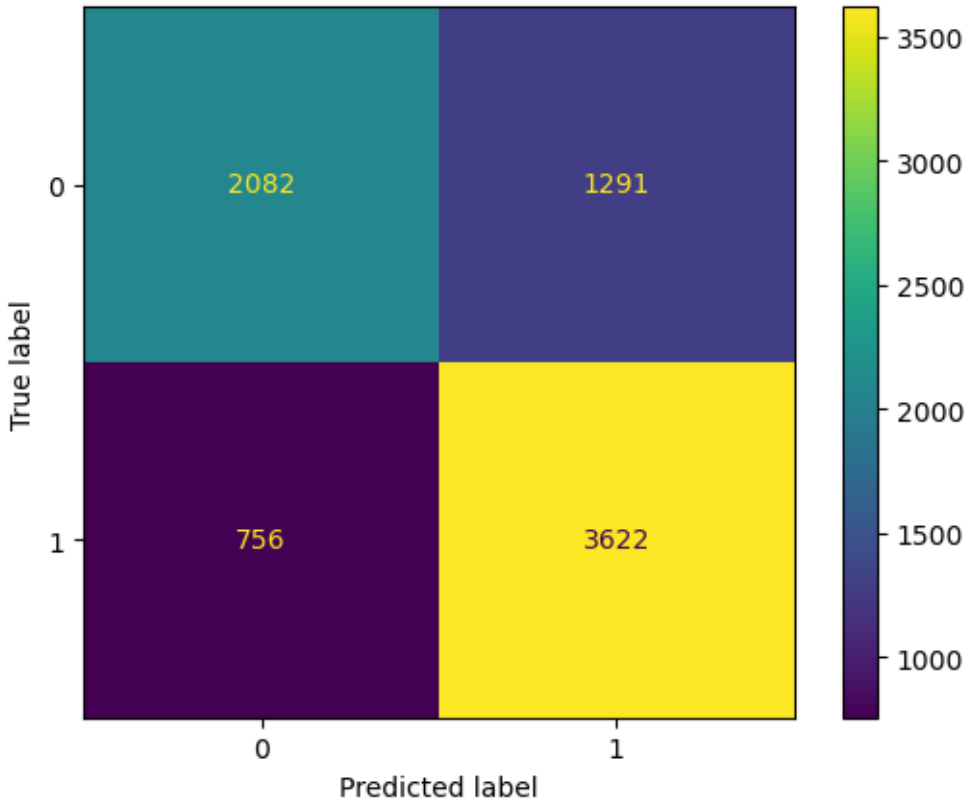
[ ]:
```
#confusion matrices of xgboost_model for train set
cm_xgboost_train=confusion_matrix(y_train, xgboost_model.predict(X_train))
ConfusionMatrixDisplay(cm_xgboost_train).plot()
```

[ ]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x29c1668afb0>
```



[ ]:
```
#confusion matrices of xgboost_model for test set
cm_xgboost_test=confusion_matrix(y_test, xgboost_model.predict(X_test))
ConfusionMatrixDisplay(cm_xgboost_test).plot()
```

[ ]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x29c160cae60>
```



3.0.8 Question 4

Based on the results obtained so far, answer the following questions, providing an explanation and trying to base your decision on multiple metrics: - Which classifiers would you choose for deployment? - Which classifier is the most "severe" (a.k.a. classifies more people as at risk of committing another crime within 3 years)? - Which classifier is the most cautious (a.k.a. classifies less people as at risk of committing another crime within 3 years)?

[ ]:
```
models = ['Logistic Regression', 'Random Forest', 'Decision Tree', 'XGBoost']
predicted_at_risk = [cm_logreg[0][1]+cm_logreg[1][1], cm_rf[0][1]+cm_rf[1][1],␣
 ↪cm_tree[0][1]+cm_tree[1][1], cm_xgboost[0][1]+cm_xgboost[1][1]]

df = pd.DataFrame({
    'Model': models,
    'Predicted as At-Risk (TP + FP)': predicted_at_risk
})

df
```

[ ]:

	Model	Predicted as At-Risk (TP + FP)
0	Logistic Regression	4146

```
1        Random Forest                          4976
2        Decision Tree                          4766
3              XGBoost                          4913
```

1. I will choose the XGBoost model for deployment because it achieves the highest accuracy, precision, recall, and F1-score among the four classifiers. This indicates that the XGBoost model outperforms the others in predictive performance and it maintains fairness across the two prediction classes, ensuring balanced outcomes for both positive and negative predictions.
2. Random Forest model is the most severe since it produces the highest number of positive predictions, which is 4976. This indicates that the Random Forest model tends to predict the positive class more frequently, which could lead to higher sensitivity but potentially more false positives.
3. Logistic Regression model is the most cautious since it produces the lowest number of positive prediction (4146).This indicates that this model is more conservative in predicting positive class, which might misses some true positive cases.

4 Part 3 : Fairness Evaluation:

Now that we have an understanding of how accurate our classifiers are across all samples, we need to measure their *fairness* across different categories. In similar problems, we are typically concerned with the classifiers being fair across different segments of protected populations (e.g. different genders or ethnicities). The original paper evaluates fairness for both gender and race, but for the purpose of this exercise we will only look at fairness across race, that is, for White and Black defendants.

4.0.1 Question 5

As we have seen in class, there is not just one fairness metric, but several, as they have different ways to identify different treatments across populations. The metrics used in the paper, which you will have to replicate, are:

1. *Predicted Positive Rate Disparity (PPRD)*, whether the numbers of positive predictions are on par across groups.
2. *Predicted Positive Group Rate Disparity (PPGRD)*, whether the rates of positive predictions are on par across groups.
3. *False Discovery Rate Disparity (FDRD)*, whether the ratios of false positives to predicted positives are on par across groups.
4. *False Positive Rate Disparity (FPRD)*, whether the ratios of false positives to actual negatives are on par across groups.
5. *False Omission Rate Disparity (FORD)*, whether the ratios of false negatives to predicted negatives are on par across groups.
6. *False Negative Rate Disparity (FNRD)*, whether the ratios of false negatives to actual positives are on par across groups.

Before jumping into code writing, we must make sure that we have a solid understanding of how these metrics are computed from the True Positive, True Negative, False Positive, and False Negative values *for each group*. We will add the subscript b and w when appropriate to identify metrics from the group of black or white defendants, respectively. Then, we will write the equations for all fairness metrics. The first one is provided to you as an example:

Metric	Formula
PPRD	(TPb + FPb) / (TPw + FPw)
PPGRD	((TPb + FPb)/ (TPb + FPb + TNb + FNb)) / ((TPw + FPw) / (TPw + FPw + TNw + FNw))
FDRD	(FPb / (TPb + FPb))/(FPw / (TPw + FPw))
FPRD	(FPb / (TNb + FPb))/(FPw / (TNw + FPw))
FORD	(FNb / (TNb + FNb))/(FNw / (TNw + FNw))
FNRD	(FNb / (TPb + FNb))/(FNw / (TPw + FNw))

Finally, the paper also computes an **Average Distance from Reference** across all the above metrics. This helps us summarizing the fairness of a classifier in a single number. Compute the Average Distance from Reference for all the classifiers, knowing that the reference is 1 (i.e. a score of 1 indicates perfect fairness). Use tha absolute value to compute the distance from the reference (e.g. a FDRD score of 0.80 and one of 1.20 both have a distance from the reference of 0.20).

Now that you have a better understanding of how to compute these metrics, do so for all the classifiers, both on the training and the test sets.

Hints: - There are several ways to write Python code to easily compute the fairness metrics we want. If you have trouble starting, talk with a TA or with the instructor during our in-class work time or office hours to come up with a plan. - Instead of copy-pasting code, it is definitely a good idea to create one or more functions to compute the fairness metrics. Writing functions in Python is very easy! If you are new to it, start here (stop before Arbitrary Keyword Arguments), and of course, come to us for more help!

[ ]:
```
# Add as many cells as needed to compute the required metrics for every␣
 ↪classifier. You may
# also add markdown cells if you want to add comments or notes about your␣
 ↪results.

def fairness_metrics(model,test_x,test_y):
  dict={}
  pred = model.predict(test_x)

  group_white = test_x["Race"] == "WHITE"
  group_black = test_x["Race"] == "BLACK"


  cm_white = confusion_matrix(test_y[group_white], pred[group_white])
  cm_black = confusion_matrix(test_y[group_black], pred[group_black])

  TNw, FPw, FNw, TPw = cm_white.ravel()
```

```
  TNb, FPb, FNb, TPb = cm_black.ravel()

  dict["PPRD"] = (TPb + FPb) / (TPw + FPw)
  dict["PPGRD"] = ((TPb + FPb)/ (TPb + FPb + TNb + FNb)) / ((TPw + FPw) /(TPw +␣
↪FPw + TNw + FNw))
  dict["FDRD"]= (FPb / (TPb + FPb))/(FPw / (TPw + FPw))
  dict["FPRD"] = (FPb / (TNb + FPb))/(FPw / (TNw + FPw))
  dict["FORD"] = (FNb / (TNb + FNb))/(FNw / (TNw + FNw))
  dict["FNRD"] = (FNb / (TPb + FNb))/(FNw / (TPw + FNw))


  AVG_D = sum(abs(value - 1) for value in dict.values()) / 6

  dict["Average Distance from Reference"] = AVG_D


  df =  pd.DataFrame(list(dict.items()), columns=['Metric', 'Value'])
  return df
```

[ ]:
```
results_dict = {
    "Logistic Regression Train Set": fairness_metrics(logreg_model, X_train,␣
↪y_train),
    "Logistic Regression Test Set": fairness_metrics(logreg_model, X_test,␣
↪y_test),
    "Random Forest Train Set": fairness_metrics(rf_model, X_train, y_train),
    "Random Forest Test Set": fairness_metrics(rf_model, X_test, y_test),
    "Decision Tree Train Set": fairness_metrics(tree_model, X_train, y_train),
    "Decision Tree Test Set": fairness_metrics(tree_model, X_test, y_test),
    "XGBoost Train Set" : fairness_metrics(xgboost_model,X_train,y_train),
    "XGBoost Test Set" : fairness_metrics(xgboost_model,X_test,y_test)
}

results_df = pd.concat(
    {name: result.set_index('Metric')['Value'] for name, result in results_dict.
↪items()}, axis=1)

results_df
```

[ ]:
```
                                  Logistic Regression Train Set  \
Metric
PPRD                                                   1.455143
PPGRD                                                  1.083916
FDRD                                                   1.060968
FPRD                                                   1.213997
FORD                                                   1.072639
FNRD                                                   0.940441
Average Distance from Reference                        0.157704
```

	Logistic Regression Test Set
Metric	
PPRD	1.469327
PPGRD	1.071104
FDRD	1.027451
FPRD	1.166160
FORD	1.091346
FNRD	0.964937
Average Distance from Reference	0.143409

	Random Forest Train Set
Metric	
PPRD	1.394653
PPGRD	1.038857
FDRD	1.792561
FPRD	1.965847
FORD	1.445670
FNRD	1.319480
Average Distance from Reference	0.492845

	Random Forest Test Set
Metric	
PPRD	1.492986
PPGRD	1.088351
FDRD	1.022388
FPRD	1.179099
FORD	1.056076
FNRD	0.869635
Average Distance from Reference	0.161544

	Decision Tree Train Set
Metric	
PPRD	1.470444
PPGRD	1.095313
FDRD	1.060007
FPRD	1.225651
FORD	1.046944
FNRD	0.873857
Average Distance from Reference	0.170750

	Decision Tree Test Set	XGBoost Train Set
Metric		
PPRD	1.505783	1.451923
PPGRD	1.097680	1.081517
FDRD	1.043059	1.156987
FPRD	1.213249	1.320936

FORD	1.086773	0.958454
FNRD	0.897645	0.812306
Average Distance from Reference	0.174817	0.206767

Metric	XGBoost Test Set
PPRD	1.434589
PPGRD	1.045781
FDRD	1.000879
FPRD	1.109142
FORD	1.132486
FNRD	1.002636
Average Distance from Reference	0.120919

4.0.2 Question 6

Based on the results obtained so far, answer the following questions, providing an explanation for each answer: - Which model exhibits the least amount of bias? - Which one is the worse? - Based on the application, which fairness metric(s) do you think should be the most important? Which one(s) could be taken less into consideration? - Finally, based on the fairness results, which model would you pick for this application?

1. The XGBoost model exhibits the least amount of bias, as it has the smallest Average Distance from Reference (around 0.121). This indicates that the XGBoost model demonstrates the most balanced performance across all fairness metrics compared to the other models, demonstrating minimal disparities between racial groups and ensuring equitable predictions.

2. The Decision Tree model performs the worst, as it has the largest Average Distance from Reference (around 0.175). This indicates that it has the poorest overall balance across the fairness metrics, resulting in a higher level of bias in its predictions, which could negatively impact certain groups.

3. FPRD (False Positive Rate Disparity) and PPGRD (Predicted Positive Group Rate Disparity) are the most important metrics:

   - FPRD: FPRD ensures that false positive rates are consistent across groups. A FPRD value close to 1 indicates that false positive predictions are fair and balanced between racial groups, minimizing the risk of disproportionately labeling individuals from certain groups as high-risk.
   - PPGRD: PPGRD ensures that positive predictions are consistent across groups. A PPGRD value close to 1 indicates that predictions are fair and representative across racial groups. Moreover, because this metric accounts for the proportional rate rather than absolute numbers in each group, it provides a more accurate assessment of fairness, even if the racial groups differ in size.

   FNRD (False Negative Rate Disparity) could be taken less into consideration, as the proportion of false negatives to actual positives is less critical in this case.

4. I would pick the XGBoost model for this application. It demonstrates the most balanced performance across key fairness metrics and has the lowest Average Distance from Reference,

this ensures that both White and Black groups receive equitable treatment in the predictions, promoting fairness and consistency.

5 Part 4: Interpretability Evaluation:

Finally, we will evaluate the *interpretability* of our models. It is important to be able to explain how the model uses each feature to make its predictions and *why* a model has given a particular response for an individual - especially important when, like in this case, people's lives are being affected.

5.0.1 Inherently Interpretable Models

Some models are known to be *inherently interpretable*, meaning we can decifer the model behavior by looking at its parameters. These models are also called "white-box" models. Logistic regression models and decision trees - in some cases - fall in this category.

5.0.2 Question 7

Run the cells below and look at the weights of the logistic regression model. For simplicity, the cells below show the 10 most positive and 10 most negative coefficients. What features bring the prediction more toward the positive class? What other features push the prediction toward the negative class? Do you see any coefficients that may be unfairly influencing the decision?

[ ]:
```
feature_names = np.array(logreg_model.named_steps['columntransformer'].
↪get_feature_names_out())
coeffs = logreg_model.named_steps["logisticregression"].coef_.flatten()
coeff_df = pd.DataFrame(coeffs, index=feature_names, columns=["Coefficient"])
coeff_df_sorted = coeff_df.sort_values(by="Coefficient", ascending=False)
```

[ ]:
```
coeff_df_sorted.head(10)
```

[ ]:

	Coefficient
pipeline-2__Gang_Affiliated_True	0.777359
pipeline-2__Age_at_Release_18-22	0.769493
pipeline-2__Delinquency_Reports_1	0.635820
pipeline-2__Age_at_Release_23-27	0.488786
pipeline-2__Prior_Arrest_Episodes_Felony_0	0.473336
pipeline-2__Gender_M	0.458249
passthrough__Prior_Revocations_Parole	0.362389
passthrough__Condition_MH_SA	0.359125
pipeline-1__Jobs_Per_Year	0.312955
pipeline-2__Prison_Years_Less than 1 year	0.307470

[ ]:
```
coeff_df_sorted.tail(10)
```

[ ]:

	Coefficient
pipeline-2__Delinquency_Reports_3	-0.205465
pipeline-2__Prior_Arrest_Episodes_PPViolationCh…	-0.216044

```
pipeline-2__Age_at_Release_38-42                    -0.235458
pipeline-2__Prior_Arrest_Episodes_Felony_2          -0.249538
pipeline-2__Age_at_Release_43-47                    -0.350816
pipeline-2__Program_Attendances_10 or more          -0.385807
pipeline-2__Prior_Arrest_Episodes_Felony_1          -0.501980
pipeline-2__Delinquency_Reports_4 or more           -0.507621
pipeline-1__Percent_Days_Employed                   -0.663675
pipeline-2__Age_at_Release_48 or older              -0.752273
```

The features like Age_at_Release_18-22(0.742), Delinquency_Reports_1(0.549), Age_at_Release_23-27(0.495), and Condition_MH_SA(0.355) bring the prediction more toward the positive class, since they have the positive coefficiet. Other features like Age_at_Release_48 or older(-0.733), Percent_Days_Employed(-0.659), Delinquency_Reports_4 or more(-0.451), rior_Arrest_Episodes_Felony_1(-0.424), and Program_Attendances_10 or more(-0.367) push the prediction toward the negative class, as they have the neagtive coefficient.

We can clearly see that there is a positive coefficient 0.266 for Gender_M, and a negative coefficient -0.266 for the Gender_F. There might be bias from the Gender. It may be unfairly influencing the decision, since the coefficient indicates the model may predict a higher risk for men than women.

5.0.3 Question 8

Now, let's look at a particular sample and try to explain its prediction. We have picked this sample because its feature values make it a hard case, one very close to the threshold between positive and negative class:

[ ]:
```
hard_sample = X_test[106:107]
hard_sample
```

[ ]:
```
     Unnamed: 0    ID Gender   Race Age_at_Release  Residence_PUMA  \
106        5645  5788      F  WHITE    48 or older               3

    Gang_Affiliated  Supervision_Risk_Score_First Supervision_Level_First  \
106             NaN                           6.0                    High

           Education_Level  … DrugTests_Cocaine_Positive  \
106  At least some college  …                         NaN

    DrugTests_Meth_Positive DrugTests_Other_Positive Percent_Days_Employed  \
106                     NaN                     NaN              0.596215

    Jobs_Per_Year Employment_Exempt Recidivism_Arrest_Year1  \
106           2.0             False                   False

    Recidivism_Arrest_Year2 Recidivism_Arrest_Year3  Training_Sample
106                   False                   False                0

[1 rows x 54 columns]
```

If you look at the ground truth for this sample (try `y_test[106:107]`) you will see that this person has not, in fact, committed a new crime within 3 years from release. But what is the prediction of the logistic regression model? Find the answer and comment below:

[ ]:
```
X_test.loc[106, 'Age_at_Release']
```

[ ]:
```
'48 or older'
```

[ ]:
```
X_test.loc[106, 'Delinquency_Reports']
```

[ ]:
```
'0'
```

[ ]:
```
X_test.loc[106, 'Condition_MH_SA']
```

[ ]:
```
True
```

[ ]:
```
# Your answer here
logreg_model.predict(hard_sample)
```

[ ]:
```
array([False])
```

Take a closer look at the feature values for this sample. What seems to have contributed the most to the final prediction? What feature pushed the most in the opposite direction?

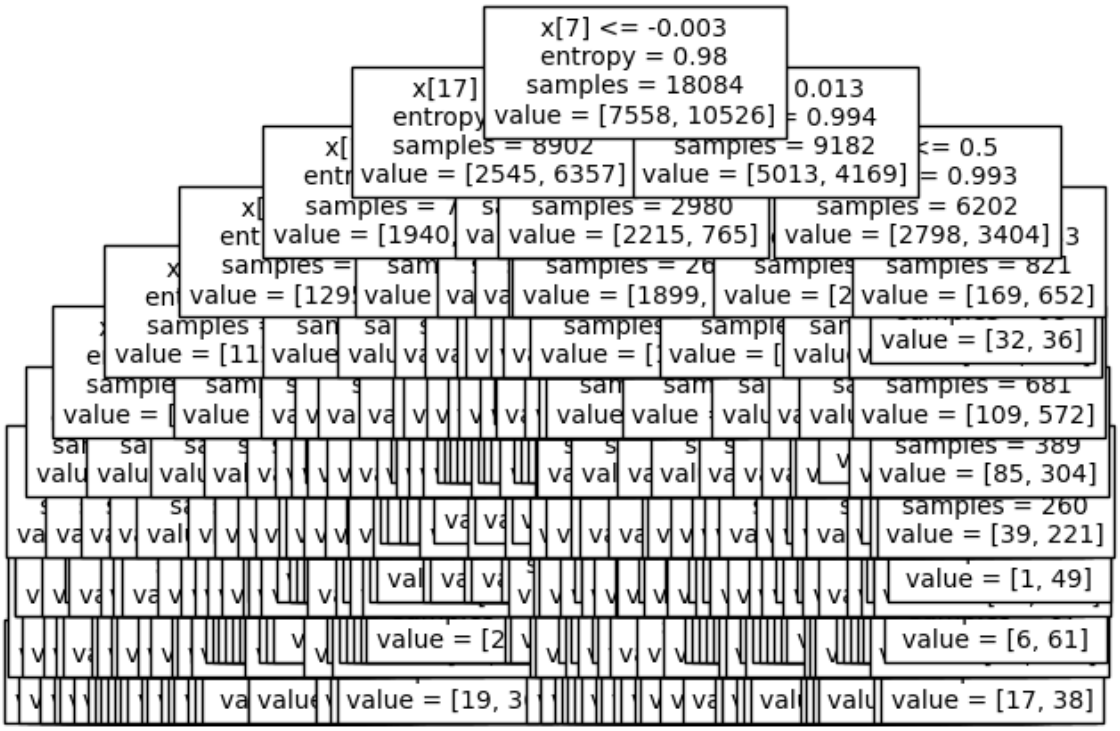
The prediction for the hard sample of the logistic regression model is False (non-recidivism), which matches the actual 'y_test[106:107]'. This person did not commit a new crime within three years. The features has negative coefficient contributed a lot to the final prediction, in partichular, Age_at_Release_48 or older(-0.733) and Percent_Days_Employed(-0.659), which are the two most negative coefficients. In this hard sample, the feature Condition_MH_SA is True. This feature has a positive coefficient (0.3549) in the logistic regression model It pushes the prediction in the opposite direction.

5.0.4 Question 9

We said that decision trees are also inherently interpretable - *potentially*. That is because, in theory, it is possible to look at the tree structure and to follow the path along the tree to see how each node influenced the decision. But this is only possible if the tree has a reasonably small size.

Run the cell below and see if you can tell what are the most influencial features in the decision tree model.

[ ]:
```
tree.plot_tree(tree_model["dt"],fontsize=10)
plt.figure(figsize=(10,6))
plt.show()
```




```
<Figure size 1000x600 with 0 Axes>
```

I would say that the features at the top of the decision tree are the most infulencial features in the decision tree model, such as X[7] Supervision_Risk_Score_First and X[17] Percent_Days_Employed. Higher a feature appears in a decision tree, more important it is, since decision tree selects the feature which provided the best split at each level.

If the method above was not satisfactory, you can try visualizing all the rules of the decision tree as text. Is this any better?

[ ]:
```
from sklearn.tree import export_text
tree_rules = export_text(tree_model.named_steps['dt'],␣
  ↪feature_names=list(tree_model.named_steps['ct'].get_feature_names_out()))
print(tree_rules)
```

```
|--- pipeline-1__Percent_Days_Employed <= -0.00
|   |--- pipeline-2__Age_at_Release_48 or older <= 0.50
|   |   |--- pipeline-2__Prior_Arrest_Episodes_PPViolationCharges_0 <= 0.50
|   |   |   |--- pipeline-2__Gang_Affiliated_True <= 0.50
|   |   |   |   |--- pipeline-2__Prior_Arrest_Episodes_Misd_6 or more <= 0.50
|   |   |   |   |   |--- pipeline-1__Supervision_Risk_Score_First <= 0.20
|   |   |   |   |   |   |--- pipeline-1__Percent_Days_Employed <= -1.15
|   |   |   |   |   |   |   |--- pipeline-2__Gender_M <= 0.50
|   |   |   |   |   |   |   |   |--- pipeline-2__Residence_Changes_0 <= 0.50
|   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |--- pipeline-2__Residence_Changes_0 >  0.50
```

```
|   |   |   |   |   |   |   |   |   |--- class: False
|   |   |   |   |   |   |   |   |--- pipeline-2__Gender_M >  0.50
|   |   |   |   |   |   |   |   |---
pipeline-2__Prior_Conviction_Episodes_Prop_0 <= 0.50
|   |   |   |   |   |   |   |   |   |---
pipeline-2__Prior_Arrest_Episodes_Misd_5 <= 0.50
|   |   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |   |---
pipeline-2__Prior_Arrest_Episodes_Misd_5 >  0.50
|   |   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |---
pipeline-2__Prior_Conviction_Episodes_Prop_0 >  0.50
|   |   |   |   |   |   |   |   |   |--- pipeline-1__Avg_Days_per_DrugTest <=
0.20
|   |   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |   |--- pipeline-1__Avg_Days_per_DrugTest >
0.20
|   |   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |--- pipeline-1__Percent_Days_Employed >  -1.15
|   |   |   |   |   |   |   |   |--- passthrough__Prior_Revocations_Parole <= 0.50
|   |   |   |   |   |   |   |   |   |--- pipeline-1__DrugTests_Meth_Positive <=
-0.12
|   |   |   |   |   |   |   |   |   |   |--- pipeline-1__Residence_PUMA <= 0.86
|   |   |   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |   |   |--- pipeline-1__Residence_PUMA >  0.86
|   |   |   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |   |--- pipeline-1__DrugTests_Meth_Positive >
-0.12
|   |   |   |   |   |   |   |   |   |--- pipeline-1__Percent_Days_Employed <=
-0.58
|   |   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |   |--- pipeline-1__Percent_Days_Employed >
-0.58
|   |   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |--- passthrough__Prior_Revocations_Parole >  0.50
|   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |--- pipeline-1__Supervision_Risk_Score_First >  0.20
|   |   |   |   |   |   |--- pipeline-2__Delinquency_Reports_4 or more <= 0.50
|   |   |   |   |   |   |   |--- pipeline-1__Avg_Days_per_DrugTest <= 0.31
|   |   |   |   |   |   |   |   |--- pipeline-2__Prior_Arrest_Episodes_Misd_0 <=
0.50
|   |   |   |   |   |   |   |   |   |--- pipeline-1__DrugTests_THC_Positive <=
0.55
|   |   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |   |--- pipeline-1__DrugTests_THC_Positive >
0.55
|   |   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |--- pipeline-2__Prior_Arrest_Episodes_Misd_0 >
```

```
0.50
|   |   |   |   |   |   |   |   |   |--- pipeline-1__DrugTests_THC_Positive <=
-0.27
|   |   |   |   |   |   |   |   |   |   |--- class: False
|   |   |   |   |   |   |   |   |   |--- pipeline-1__DrugTests_THC_Positive >
-0.27
|   |   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |--- pipeline-1__Avg_Days_per_DrugTest >  0.31
|   |   |   |   |   |   |   |   |---
pipeline-2__Prior_Conviction_Episodes_Drug_2 or more <= 0.50
|   |   |   |   |   |   |   |   |   |--- pipeline-1__Residence_PUMA <= 0.72
|   |   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |   |--- pipeline-1__Residence_PUMA >  0.72
|   |   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |---
pipeline-2__Prior_Conviction_Episodes_Drug_2 or more >  0.50
|   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |--- pipeline-2__Delinquency_Reports_4 or more >  0.50
|   |   |   |   |   |   |   |--- pipeline-1__DrugTests_THC_Positive <= 1.07
|   |   |   |   |   |   |   |---
passthrough__Prior_Conviction_Episodes_PPViolationCharges <= 0.50
|   |   |   |   |   |   |   |   |---
pipeline-2__Prior_Arrest_Episodes_Drug_0 <= 0.50
|   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |---
pipeline-2__Prior_Arrest_Episodes_Drug_0 >  0.50
|   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |---
passthrough__Prior_Conviction_Episodes_PPViolationCharges >  0.50
|   |   |   |   |   |   |   |   |   |--- class: False
|   |   |   |   |   |   |   |--- pipeline-1__DrugTests_THC_Positive >  1.07
|   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |--- pipeline-2__Prior_Arrest_Episodes_Misd_6 or more >  0.50
|   |   |   |   |   |--- pipeline-2__Delinquency_Reports_4 or more <= 0.50
|   |   |   |   |   |   |--- pipeline-1__Percent_Days_Employed <= -1.13
|   |   |   |   |   |   |   |--- pipeline-2__Prior_Conviction_Episodes_Prop_3 or
more <= 0.50
|   |   |   |   |   |   |   |   |--- pipeline-2__Age_at_Release_28-32 <= 0.50
|   |   |   |   |   |   |   |   |   |--- pipeline-2__Prison_Years_Less than 1
year <= 0.50
|   |   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |   |--- pipeline-2__Prison_Years_Less than 1
year >  0.50
|   |   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |--- pipeline-2__Age_at_Release_28-32 >  0.50
|   |   |   |   |   |   |   |   |   |--- pipeline-1__Residence_PUMA <= -0.12
|   |   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |   |--- pipeline-1__Residence_PUMA >  -0.12
```

```
|   |   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |---  pipeline-2__Prior_Conviction_Episodes_Prop_3 or
more >  0.50
|   |   |   |   |   |   |   |   |   |--- pipeline-2__Residence_Changes_1 <= 0.50
|   |   |   |   |   |   |   |   |   |---
passthrough__Prior_Revocations_Probation <= 0.50
|   |   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |   |---
passthrough__Prior_Revocations_Probation >  0.50
|   |   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |   |--- pipeline-2__Residence_Changes_1 >  0.50
|   |   |   |   |   |   |   |   |   |   |--- pipeline-1__Residence_PUMA <= -0.12
|   |   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |   |--- pipeline-1__Residence_PUMA >  -0.12
|   |   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |--- pipeline-1__Percent_Days_Employed >  -1.13
|   |   |   |   |   |   |   |   |--- pipeline-1__DrugTests_Meth_Positive <= -0.12
|   |   |   |   |   |   |   |   |---
pipeline-2__Supervision_Level_First_Specialized <= 0.50
|   |   |   |   |   |   |   |   |   |--- pipeline-1__Avg_Days_per_DrugTest <=
0.68
|   |   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |   |--- pipeline-1__Avg_Days_per_DrugTest >
0.68
|   |   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |---
pipeline-2__Supervision_Level_First_Specialized >  0.50
|   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |--- pipeline-1__DrugTests_Meth_Positive >  -0.12
|   |   |   |   |   |   |   |   |   |--- pipeline-1__Residence_PUMA <= 0.72
|   |   |   |   |   |   |   |   |   |---
pipeline-1__Supervision_Risk_Score_First <= -0.24
|   |   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |   |---
pipeline-1__Supervision_Risk_Score_First >  -0.24
|   |   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |   |--- pipeline-1__Residence_PUMA >  0.72
|   |   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |--- pipeline-2__Delinquency_Reports_4 or more >  0.50
|   |   |   |   |   |   |   |--- pipeline-1__Residence_PUMA <= -1.10
|   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |--- pipeline-1__Residence_PUMA >  -1.10
|   |   |   |   |   |   |   |   |--- pipeline-1__DrugTests_THC_Positive <= -0.00
|   |   |   |   |   |   |   |   |   |--- pipeline-2__Prison_Offense_Property <= 0.50
|   |   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |   |--- pipeline-2__Prison_Offense_Property >  0.50
|   |   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |--- pipeline-1__DrugTests_THC_Positive >  -0.00
```

```
|   |   |   |   |   |   |   |   |--- pipeline-2__Education_Level_Less than HS
diploma <= 0.50
|   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |--- pipeline-2__Education_Level_Less than HS
diploma >  0.50
|   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |--- pipeline-2__Gang_Affiliated_True >  0.50
|   |   |   |   |--- pipeline-2__Delinquency_Reports_4 or more <= 0.50
|   |   |   |   |   |--- pipeline-2__Race_WHITE <= 0.50
|   |   |   |   |   |   |--- pipeline-1__Avg_Days_per_DrugTest <= -0.54
|   |   |   |   |   |   |   |--- pipeline-1__Avg_Days_per_DrugTest <= -0.68
|   |   |   |   |   |   |   |   |--- passthrough__Prior_Conviction_Episodes_Viol
<= 0.50
|   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |--- passthrough__Prior_Conviction_Episodes_Viol
>  0.50
|   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |--- pipeline-1__Avg_Days_per_DrugTest >  -0.68
|   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |--- pipeline-1__Avg_Days_per_DrugTest >  -0.54
|   |   |   |   |   |   |   |--- pipeline-1__Avg_Days_per_DrugTest <= 0.00
|   |   |   |   |   |   |   |   |---
pipeline-2__Prior_Arrest_Episodes_Property_5 or more <= 0.50
|   |   |   |   |   |   |   |   |   |--- pipeline-1__DrugTests_THC_Positive <=
-0.08
|   |   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |   |--- pipeline-1__DrugTests_THC_Positive >
-0.08
|   |   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |---
pipeline-2__Prior_Arrest_Episodes_Property_5 or more >  0.50
|   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |--- pipeline-1__Avg_Days_per_DrugTest >  0.00
|   |   |   |   |   |   |   |   |--- pipeline-1__Avg_Days_per_DrugTest <= 0.75
|   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |--- pipeline-1__Avg_Days_per_DrugTest >  0.75
|   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |--- pipeline-2__Race_WHITE >  0.50
|   |   |   |   |   |   |--- pipeline-2__Residence_Changes_1 <= 0.50
|   |   |   |   |   |   |   |--- pipeline-2__Education_Level_Less than HS
diploma <= 0.50
|   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |--- pipeline-2__Education_Level_Less than HS
diploma >  0.50
|   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |--- pipeline-2__Residence_Changes_1 >  0.50
|   |   |   |   |   |   |   |--- class: True
|   |   |   |   |--- pipeline-2__Delinquency_Reports_4 or more >  0.50
```

```
|   |   |   |   |   |--- pipeline-2__Prior_Conviction_Episodes_Prop_3 or more <=
0.50
|   |   |   |   |   |   |--- pipeline-1__Avg_Days_per_DrugTest <= -0.16
|   |   |   |   |   |   |   |--- pipeline-1__DrugTests_THC_Positive <= 0.63
|   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |--- pipeline-1__DrugTests_THC_Positive >  0.63
|   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |--- pipeline-1__Avg_Days_per_DrugTest >  -0.16
|   |   |   |   |   |   |   |--- pipeline-1__Avg_Days_per_DrugTest <= 0.15
|   |   |   |   |   |   |   |   |--- class: False
|   |   |   |   |   |   |   |--- pipeline-1__Avg_Days_per_DrugTest >  0.15
|   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |--- pipeline-2__Prior_Conviction_Episodes_Prop_3 or more >
0.50
|   |   |   |   |   |   |--- class: True
|   |   |--- pipeline-2__Prior_Arrest_Episodes_PPViolationCharges_0 >  0.50
|   |   |   |--- pipeline-2__Gang_Affiliated_True <= 0.50
|   |   |   |   |--- pipeline-1__Supervision_Risk_Score_First <= 0.20
|   |   |   |   |   |--- pipeline-1__Percent_Days_Employed <= -1.15
|   |   |   |   |   |   |--- pipeline-1__Jobs_Per_Year <= -0.77
|   |   |   |   |   |   |   |--- pipeline-2__Gender_M <= 0.50
|   |   |   |   |   |   |   |   |--- class: False
|   |   |   |   |   |   |   |--- pipeline-2__Gender_M >  0.50
|   |   |   |   |   |   |   |   |--- pipeline-2__Age_at_Release_43-47 <= 0.50
|   |   |   |   |   |   |   |   |   |---
pipeline-2__Prior_Arrest_Episodes_Violent_0 <= 0.50
|   |   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |   |---
pipeline-2__Prior_Arrest_Episodes_Violent_0 >  0.50
|   |   |   |   |   |   |   |   |   |   |--- class: False
|   |   |   |   |   |   |   |   |--- pipeline-2__Age_at_Release_43-47 >  0.50
|   |   |   |   |   |   |   |   |   |--- class: False
|   |   |   |   |   |   |--- pipeline-1__Jobs_Per_Year >  -0.77
|   |   |   |   |   |   |   |--- class: False
|   |   |   |   |   |--- pipeline-1__Percent_Days_Employed >  -1.15
|   |   |   |   |   |   |--- pipeline-1__Avg_Days_per_DrugTest <= 0.55
|   |   |   |   |   |   |   |--- pipeline-1__DrugTests_Cocaine_Positive <= -0.12
|   |   |   |   |   |   |   |   |--- pipeline-2__Prior_Arrest_Episodes_Felony_1
<= 0.50
|   |   |   |   |   |   |   |   |   |---
pipeline-2__Prior_Arrest_Episodes_Property_0 <= 0.50
|   |   |   |   |   |   |   |   |   |   |--- class: False
|   |   |   |   |   |   |   |   |   |---
pipeline-2__Prior_Arrest_Episodes_Property_0 >  0.50
|   |   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |--- pipeline-2__Prior_Arrest_Episodes_Felony_1
>  0.50
|   |   |   |   |   |   |   |   |   |--- class: False
```

```
|   |   |   |   |   |   |   |   |--- pipeline-1__DrugTests_Cocaine_Positive >  -0.12
|   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |--- pipeline-1__Avg_Days_per_DrugTest >  0.55
|   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |--- pipeline-1__Supervision_Risk_Score_First >  0.20
|   |   |   |   |   |   |--- pipeline-1__DrugTests_THC_Positive <= 0.46
|   |   |   |   |   |   |   |--- pipeline-1__Jobs_Per_Year <= -0.44
|   |   |   |   |   |   |   |   |--- pipeline-2__Residence_Changes_0 <= 0.50
|   |   |   |   |   |   |   |   |   |--- pipeline-2__Prison_Offense_Property <= 0.50
|   |   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |   |--- pipeline-2__Prison_Offense_Property >  0.50
|   |   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |--- pipeline-2__Residence_Changes_0 >  0.50
|   |   |   |   |   |   |   |   |   |--- passthrough__Condition_MH_SA <= 0.50
|   |   |   |   |   |   |   |   |   |   |--- class: False
|   |   |   |   |   |   |   |   |   |--- passthrough__Condition_MH_SA >  0.50
|   |   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |--- pipeline-1__Jobs_Per_Year >  -0.44
|   |   |   |   |   |   |   |   |--- pipeline-1__Jobs_Per_Year <= 0.13
|   |   |   |   |   |   |   |   |   |--- class: False
|   |   |   |   |   |   |   |   |--- pipeline-1__Jobs_Per_Year >  0.13
|   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |--- pipeline-1__DrugTests_THC_Positive >  0.46
|   |   |   |   |   |   |   |--- pipeline-1__Residence_PUMA <= -0.12
|   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |--- pipeline-1__Residence_PUMA >  -0.12
|   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |--- pipeline-2__Gang_Affiliated_True >  0.50
|   |   |   |   |   |--- pipeline-1__Supervision_Risk_Score_First <= -0.02
|   |   |   |   |   |   |--- pipeline-1__Avg_Days_per_DrugTest <= -0.01
|   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |--- pipeline-1__Avg_Days_per_DrugTest >  -0.01
|   |   |   |   |   |   |   |--- class: True
|   |   |   |   |--- pipeline-1__Supervision_Risk_Score_First >  -0.02
|   |   |   |   |   |--- pipeline-2__Delinquency_Reports_0 <= 0.50
|   |   |   |   |   |   |--- pipeline-1__Residence_PUMA <= -0.54
|   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |--- pipeline-1__Residence_PUMA >  -0.54
|   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |--- pipeline-2__Delinquency_Reports_0 >  0.50
|   |   |   |   |   |   |--- pipeline-2__Prior_Arrest_Episodes_Drug_0 <= 0.50
|   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |--- pipeline-2__Prior_Arrest_Episodes_Drug_0 >  0.50
|   |   |   |   |   |   |   |--- class: True
|   |--- pipeline-2__Age_at_Release_48 or older >  0.50
|   |   |--- pipeline-2__Prior_Conviction_Episodes_Prop_3 or more <= 0.50
|   |   |   |--- pipeline-2__Prior_Arrest_Episodes_Misd_6 or more <= 0.50
|   |   |   |   |--- pipeline-1__Percent_Days_Employed <= -1.15
```

```
|   |   |   |   |   |--- pipeline-1__Jobs_Per_Year <= -0.82
|   |   |   |   |   |   |--- pipeline-2__Prior_Arrest_Episodes_Property_0 <=
0.50
|   |   |   |   |   |   |   |--- pipeline-1__Avg_Days_per_DrugTest <= -0.05
|   |   |   |   |   |   |   |   |--- class: False
|   |   |   |   |   |   |   |--- pipeline-1__Avg_Days_per_DrugTest >  -0.05
|   |   |   |   |   |   |   |   |--- class: False
|   |   |   |   |   |   |--- pipeline-2__Prior_Arrest_Episodes_Property_0 >
0.50
|   |   |   |   |   |   |   |--- class: False
|   |   |   |   |   |--- pipeline-1__Jobs_Per_Year >  -0.82
|   |   |   |   |   |   |--- class: False
|   |   |   |   |--- pipeline-1__Percent_Days_Employed >  -1.15
|   |   |   |   |   |--- pipeline-2__Prior_Arrest_Episodes_PPViolationCharges_0
<= 0.50
|   |   |   |   |   |   |--- class: True
|   |   |   |   |   |--- pipeline-2__Prior_Arrest_Episodes_PPViolationCharges_0
>  0.50
|   |   |   |   |   |   |--- class: False
|   |   |   |--- pipeline-2__Prior_Arrest_Episodes_Misd_6 or more >  0.50
|   |   |   |   |--- pipeline-1__Avg_Days_per_DrugTest <= 0.00
|   |   |   |   |   |--- pipeline-1__Residence_PUMA <= 0.16
|   |   |   |   |   |   |--- class: False
|   |   |   |   |   |--- pipeline-1__Residence_PUMA >  0.16
|   |   |   |   |   |   |--- class: True
|   |   |   |   |--- pipeline-1__Avg_Days_per_DrugTest >  0.00
|   |   |   |   |   |--- class: True
|   |   |--- pipeline-2__Prior_Conviction_Episodes_Prop_3 or more >  0.50
|   |   |   |--- pipeline-1__Percent_Days_Employed <= -1.14
|   |   |   |   |--- pipeline-1__DrugTests_THC_Positive <= -0.20
|   |   |   |   |   |--- pipeline-1__Avg_Days_per_DrugTest <= 0.15
|   |   |   |   |   |   |--- pipeline-2__Race_WHITE <= 0.50
|   |   |   |   |   |   |   |--- pipeline-2__Prior_Conviction_Episodes_Misd_4 or
more <= 0.50
|   |   |   |   |   |   |   |   |--- class: False
|   |   |   |   |   |   |   |--- pipeline-2__Prior_Conviction_Episodes_Misd_4 or
more >  0.50
|   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |--- pipeline-2__Race_WHITE >  0.50
|   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |--- pipeline-1__Avg_Days_per_DrugTest >  0.15
|   |   |   |   |   |   |--- class: True
|   |   |   |   |--- pipeline-1__DrugTests_THC_Positive >  -0.20
|   |   |   |   |   |--- pipeline-2__Prior_Conviction_Episodes_Misd_4 or more <=
0.50
|   |   |   |   |   |   |--- class: True
|   |   |   |   |   |--- pipeline-2__Prior_Conviction_Episodes_Misd_4 or more >
0.50
```

```
|   |   |   |   |   |   |--- class: True
|   |   |   |--- pipeline-1__Percent_Days_Employed >  -1.14
|   |   |   |   |--- pipeline-1__Jobs_Per_Year <= 0.03
|   |   |   |   |   |--- class: True
|   |   |   |   |--- pipeline-1__Jobs_Per_Year >  0.03
|   |   |   |   |   |--- class: True
|--- pipeline-1__Percent_Days_Employed >  -0.00
|   |--- pipeline-1__Jobs_Per_Year <= 0.01
|   |   |--- pipeline-1__Percent_Days_Employed <= 0.00
|   |   |   |--- pipeline-1__DrugTests_THC_Positive <= -0.00
|   |   |   |   |--- class: False
|   |   |   |--- pipeline-1__DrugTests_THC_Positive >  -0.00
|   |   |   |   |--- class: False
|   |   |--- pipeline-1__Percent_Days_Employed >  0.00
|   |   |   |--- pipeline-1__Percent_Days_Employed <= 0.39
|   |   |   |   |--- pipeline-2__Prior_Arrest_Episodes_Property_0 <= 0.50
|   |   |   |   |   |--- pipeline-1__DrugTests_Cocaine_Positive <= -0.12
|   |   |   |   |   |   |---
pipeline-2__Prior_Arrest_Episodes_PPViolationCharges_5 or more <= 0.50
|   |   |   |   |   |   |   |--- pipeline-1__Supervision_Risk_Score_First <=
-0.24
|   |   |   |   |   |   |   |   |--- class: False
|   |   |   |   |   |   |   |--- pipeline-1__Supervision_Risk_Score_First >
-0.24
|   |   |   |   |   |   |   |   |--- pipeline-1__Percent_Days_Employed <= 0.18
|   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |--- pipeline-1__Percent_Days_Employed >  0.18
|   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |---
pipeline-2__Prior_Arrest_Episodes_PPViolationCharges_5 or more >  0.50
|   |   |   |   |   |   |--- class: True
|   |   |   |   |   |--- pipeline-1__DrugTests_Cocaine_Positive >  -0.12
|   |   |   |   |   |   |--- class: True
|   |   |   |   |--- pipeline-2__Prior_Arrest_Episodes_Property_0 >  0.50
|   |   |   |   |   |--- pipeline-1__Percent_Days_Employed <= 0.26
|   |   |   |   |   |   |--- class: True
|   |   |   |   |   |--- pipeline-1__Percent_Days_Employed >  0.26
|   |   |   |   |   |   |--- class: False
|   |   |   |--- pipeline-1__Percent_Days_Employed >  0.39
|   |   |   |   |--- pipeline-1__Jobs_Per_Year <= -0.57
|   |   |   |   |   |--- pipeline-1__Percent_Days_Employed <= 0.95
|   |   |   |   |   |   |--- pipeline-1__Jobs_Per_Year <= -0.63
|   |   |   |   |   |   |   |--- class: False
|   |   |   |   |   |   |--- pipeline-1__Jobs_Per_Year >  -0.63
|   |   |   |   |   |   |   |--- class: False
|   |   |   |   |   |--- pipeline-1__Percent_Days_Employed >  0.95
|   |   |   |   |   |   |--- class: False
|   |   |   |   |--- pipeline-1__Jobs_Per_Year >  -0.57
```

```
|   |   |   |   |   |--- pipeline-1__Percent_Days_Employed <= 1.10
|   |   |   |   |   |   |--- pipeline-1__Jobs_Per_Year <= -0.36
|   |   |   |   |   |   |   |--- pipeline-1__DrugTests_Meth_Positive <= -0.07
|   |   |   |   |   |   |   |   |--- pipeline-2__Prior_Arrest_Episodes_Felony_10
or more <= 0.50
|   |   |   |   |   |   |   |   |   |--- pipeline-1__DrugTests_THC_Positive <=
-0.14
|   |   |   |   |   |   |   |   |   |   |--- class: False
|   |   |   |   |   |   |   |   |   |--- pipeline-1__DrugTests_THC_Positive >
-0.14
|   |   |   |   |   |   |   |   |   |   |--- class: False
|   |   |   |   |   |   |   |   |--- pipeline-2__Prior_Arrest_Episodes_Felony_10
or more >  0.50
|   |   |   |   |   |   |   |   |   |--- class: False
|   |   |   |   |   |   |   |--- pipeline-1__DrugTests_Meth_Positive >  -0.07
|   |   |   |   |   |   |   |   |--- class: False
|   |   |   |   |   |   |--- pipeline-1__Jobs_Per_Year >  -0.36
|   |   |   |   |   |   |   |--- pipeline-2__Prior_Arrest_Episodes_Property_0 <=
0.50
|   |   |   |   |   |   |   |   |--- pipeline-2__Age_at_Release_48 or older <=
0.50
|   |   |   |   |   |   |   |   |   |--- pipeline-1__Jobs_Per_Year <= -0.16
|   |   |   |   |   |   |   |   |   |   |--- class: False
|   |   |   |   |   |   |   |   |   |--- pipeline-1__Jobs_Per_Year >  -0.16
|   |   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |--- pipeline-2__Age_at_Release_48 or older >
0.50
|   |   |   |   |   |   |   |   |   |--- class: False
|   |   |   |   |   |   |   |--- pipeline-2__Prior_Arrest_Episodes_Property_0 >
0.50
|   |   |   |   |   |   |   |   |--- passthrough__Condition_Cog_Ed <= 0.50
|   |   |   |   |   |   |   |   |   |--- pipeline-1__Jobs_Per_Year <= -0.18
|   |   |   |   |   |   |   |   |   |   |--- class: False
|   |   |   |   |   |   |   |   |   |--- pipeline-1__Jobs_Per_Year >  -0.18
|   |   |   |   |   |   |   |   |   |   |--- class: False
|   |   |   |   |   |   |   |   |--- passthrough__Condition_Cog_Ed >  0.50
|   |   |   |   |   |   |   |   |   |--- class: False
|   |   |   |   |   |--- pipeline-1__Percent_Days_Employed >  1.10
|   |   |   |   |   |   |--- pipeline-1__Jobs_Per_Year <= -0.45
|   |   |   |   |   |   |   |--- pipeline-1__Residence_PUMA <= -0.40
|   |   |   |   |   |   |   |   |--- class: False
|   |   |   |   |   |   |   |--- pipeline-1__Residence_PUMA >  -0.40
|   |   |   |   |   |   |   |   |--- class: False
|   |   |   |   |   |   |--- pipeline-1__Jobs_Per_Year >  -0.45
|   |   |   |   |   |   |   |--- pipeline-2__Prior_Arrest_Episodes_Misd_6 or
more <= 0.50
|   |   |   |   |   |   |   |   |--- pipeline-2__Program_Attendances_10 or more
<= 0.50
```

```
|   |   |   |   |   |   |   |   |   |--- pipeline-2__Dependents_3 or more <=
0.50
|   |   |   |   |   |   |   |   |   |   |--- class: False
|   |   |   |   |   |   |   |   |   |--- pipeline-2__Dependents_3 or more >
0.50
|   |   |   |   |   |   |   |   |   |   |--- class: False
|   |   |   |   |   |   |   |   |   |--- pipeline-2__Program_Attendances_10 or more
>  0.50
|   |   |   |   |   |   |   |   |   |--- class: False
|   |   |   |   |   |   |   |   |--- pipeline-2__Prior_Arrest_Episodes_Misd_6 or
more >  0.50
|   |   |   |   |   |   |   |   |--- pipeline-2__Residence_Changes_1 <= 0.50
|   |   |   |   |   |   |   |   |   |---
pipeline-1__Supervision_Risk_Score_First <= -0.24
|   |   |   |   |   |   |   |   |   |   |--- class: False
|   |   |   |   |   |   |   |   |   |---
pipeline-1__Supervision_Risk_Score_First >  -0.24
|   |   |   |   |   |   |   |   |   |   |--- class: False
|   |   |   |   |   |   |   |   |   |--- pipeline-2__Residence_Changes_1 >  0.50
|   |   |   |   |   |   |   |   |   |--- class: False
|   |--- pipeline-1__Jobs_Per_Year >  0.01
|   |   |--- pipeline-2__Gang_Affiliated_True <= 0.50
|   |   |   |--- pipeline-2__Prior_Arrest_Episodes_PPViolationCharges_0 <= 0.50
|   |   |   |   |--- pipeline-2__Prior_Arrest_Episodes_PPViolationCharges_5 or
more <= 0.50
|   |   |   |   |   |--- pipeline-1__Percent_Days_Employed <= 0.92
|   |   |   |   |   |   |--- pipeline-2__Program_Attendances_10 or more <= 0.50
|   |   |   |   |   |   |   |--- pipeline-1__Supervision_Risk_Score_First <=
-0.67
|   |   |   |   |   |   |   |   |--- passthrough__Condition_MH_SA <= 0.50
|   |   |   |   |   |   |   |   |   |--- class: False
|   |   |   |   |   |   |   |   |--- passthrough__Condition_MH_SA >  0.50
|   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |--- pipeline-1__Supervision_Risk_Score_First >
-0.67
|   |   |   |   |   |   |   |   |--- pipeline-1__DrugTests_Cocaine_Positive <=
-0.12
|   |   |   |   |   |   |   |   |   |---
pipeline-2__Prior_Conviction_Episodes_Drug_0 <= 0.50
|   |   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |   |---
pipeline-2__Prior_Conviction_Episodes_Drug_0 >  0.50
|   |   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |   |--- pipeline-1__DrugTests_Cocaine_Positive >
-0.12
|   |   |   |   |   |   |   |   |   |---
pipeline-2__Prior_Conviction_Episodes_Drug_0 <= 0.50
|   |   |   |   |   |   |   |   |   |   |--- class: True
```

```
|   |   |   |   |   |   |   |   |   |---
pipeline-2__Prior_Conviction_Episodes_Drug_0 >  0.50
|   |   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |--- pipeline-2__Program_Attendances_10 or more >  0.50
|   |   |   |   |   |   |   |   |--- pipeline-1__Residence_PUMA <= -0.12
|   |   |   |   |   |   |   |   |   |--- class: False
|   |   |   |   |   |   |   |   |--- pipeline-1__Residence_PUMA >  -0.12
|   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |--- pipeline-1__Percent_Days_Employed >  0.92
|   |   |   |   |   |   |   |---
pipeline-2__Prior_Arrest_Episodes_PPViolationCharges_1 <= 0.50
|   |   |   |   |   |   |   |   |--- pipeline-1__DrugTests_Meth_Positive <= 1.31
|   |   |   |   |   |   |   |   |   |--- pipeline-2__Residence_Changes_3 or more <=
0.50
|   |   |   |   |   |   |   |   |   |   |--- pipeline-1__Avg_Days_per_DrugTest <=
-0.76
|   |   |   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |   |   |--- pipeline-1__Avg_Days_per_DrugTest >
-0.76
|   |   |   |   |   |   |   |   |   |   |   |--- class: False
|   |   |   |   |   |   |   |   |   |--- pipeline-2__Residence_Changes_3 or more >
0.50
|   |   |   |   |   |   |   |   |   |   |--- pipeline-1__DrugTests_THC_Positive <=
-0.42
|   |   |   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |   |   |--- pipeline-1__DrugTests_THC_Positive >
-0.42
|   |   |   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |--- pipeline-1__DrugTests_Meth_Positive >  1.31
|   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |---
pipeline-2__Prior_Arrest_Episodes_PPViolationCharges_1 >  0.50
|   |   |   |   |   |   |   |   |--- pipeline-2__Age_at_Release_48 or older <= 0.50
|   |   |   |   |   |   |   |   |   |---
pipeline-2__Prior_Conviction_Episodes_Prop_3 or more <= 0.50
|   |   |   |   |   |   |   |   |   |   |--- pipeline-1__DrugTests_THC_Positive <=
-0.27
|   |   |   |   |   |   |   |   |   |   |   |--- class: False
|   |   |   |   |   |   |   |   |   |   |--- pipeline-1__DrugTests_THC_Positive >
-0.27
|   |   |   |   |   |   |   |   |   |   |   |--- class: False
|   |   |   |   |   |   |   |   |   |---
pipeline-2__Prior_Conviction_Episodes_Prop_3 or more >  0.50
|   |   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |--- pipeline-2__Age_at_Release_48 or older >  0.50
|   |   |   |   |   |   |   |   |   |--- class: False
|   |   |   |   |--- pipeline-2__Prior_Arrest_Episodes_PPViolationCharges_5 or
more >  0.50
```

```
|   |   |   |   |   |--- pipeline-1__Percent_Days_Employed <= 0.96
|   |   |   |   |   |   |--- pipeline-1__Avg_Days_per_DrugTest <= -0.20
|   |   |   |   |   |   |   |--- pipeline-1__Percent_Days_Employed <= 0.65
|   |   |   |   |   |   |   |   |--- pipeline-1__Avg_Days_per_DrugTest <= -0.65
|   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |--- pipeline-1__Avg_Days_per_DrugTest >  -0.65
|   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |--- pipeline-1__Percent_Days_Employed >  0.65
|   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |--- pipeline-1__Avg_Days_per_DrugTest >  -0.20
|   |   |   |   |   |   |   |--- pipeline-2__Dependents_3 or more <= 0.50
|   |   |   |   |   |   |   |   |--- pipeline-1__Supervision_Risk_Score_First <=
-0.24
|   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |--- pipeline-1__Supervision_Risk_Score_First >
-0.24
|   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |--- pipeline-2__Dependents_3 or more >  0.50
|   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |--- pipeline-1__Percent_Days_Employed >  0.96
|   |   |   |   |   |   |--- pipeline-1__Jobs_Per_Year <= 1.53
|   |   |   |   |   |   |   |--- pipeline-1__Jobs_Per_Year <= 0.31
|   |   |   |   |   |   |   |   |--- pipeline-1__Jobs_Per_Year <= 0.28
|   |   |   |   |   |   |   |   |   |--- pipeline-2__Prison_Offense_Property <=
0.50
|   |   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |   |--- pipeline-2__Prison_Offense_Property >
0.50
|   |   |   |   |   |   |   |   |   |   |--- class: False
|   |   |   |   |   |   |   |   |--- pipeline-1__Jobs_Per_Year >  0.28
|   |   |   |   |   |   |   |   |   |---
pipeline-1__Supervision_Risk_Score_First <= 0.60
|   |   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |   |---
pipeline-1__Supervision_Risk_Score_First >  0.60
|   |   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |--- pipeline-1__Jobs_Per_Year >  0.31
|   |   |   |   |   |   |   |   |--- pipeline-1__Jobs_Per_Year <= 0.74
|   |   |   |   |   |   |   |   |   |--- class: False
|   |   |   |   |   |   |   |   |--- pipeline-1__Jobs_Per_Year >  0.74
|   |   |   |   |   |   |   |   |   |--- pipeline-2__Prison_Offense_Property <=
0.50
|   |   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |   |--- pipeline-2__Prison_Offense_Property >
0.50
|   |   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |--- pipeline-1__Jobs_Per_Year >  1.53
|   |   |   |   |   |   |   |--- pipeline-1__Jobs_Per_Year <= 1.55
```

```
|   |   |   |   |   |   |   |   |--- pipeline-1__Avg_Days_per_DrugTest <= -0.48
|   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |--- pipeline-1__Avg_Days_per_DrugTest >  -0.48
|   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |--- pipeline-1__Jobs_Per_Year >  1.55
|   |   |   |   |   |   |   |   |--- passthrough__Condition_Cog_Ed <= 0.50
|   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |--- passthrough__Condition_Cog_Ed >  0.50
|   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |--- pipeline-2__Prior_Arrest_Episodes_PPViolationCharges_0 >  0.50
|   |   |   |   |--- pipeline-1__Percent_Days_Employed <= 1.00
|   |   |   |   |   |--- pipeline-2__Prior_Arrest_Episodes_Felony_1 <= 0.50
|   |   |   |   |   |   |--- pipeline-1__Supervision_Risk_Score_First <= -0.67
|   |   |   |   |   |   |   |--- class: False
|   |   |   |   |   |   |--- pipeline-1__Supervision_Risk_Score_First >  -0.67
|   |   |   |   |   |   |   |--- pipeline-2__Prior_Conviction_Episodes_Drug_0 <=
0.50
|   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |--- pipeline-2__Prior_Conviction_Episodes_Drug_0 >
0.50
|   |   |   |   |   |   |   |   |--- passthrough__Condition_Cog_Ed <= 0.50
|   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |--- passthrough__Condition_Cog_Ed >  0.50
|   |   |   |   |   |   |   |   |   |--- class: False
|   |   |   |   |   |--- pipeline-2__Prior_Arrest_Episodes_Felony_1 >  0.50
|   |   |   |   |   |   |--- pipeline-2__Prior_Arrest_Episodes_Misd_0 <= 0.50
|   |   |   |   |   |   |   |--- class: False
|   |   |   |   |   |   |--- pipeline-2__Prior_Arrest_Episodes_Misd_0 >  0.50
|   |   |   |   |   |   |   |--- class: False
|   |   |   |   |--- pipeline-1__Percent_Days_Employed >  1.00
|   |   |   |   |   |--- pipeline-2__Prison_Offense_Violent/Sex <= 0.50
|   |   |   |   |   |   |--- pipeline-2__Gender_M <= 0.50
|   |   |   |   |   |   |   |--- passthrough__Condition_Cog_Ed <= 0.50
|   |   |   |   |   |   |   |   |--- class: False
|   |   |   |   |   |   |   |--- passthrough__Condition_Cog_Ed >  0.50
|   |   |   |   |   |   |   |   |--- class: False
|   |   |   |   |   |   |--- pipeline-2__Gender_M >  0.50
|   |   |   |   |   |   |   |--- pipeline-1__Supervision_Risk_Score_First <=
-0.67
|   |   |   |   |   |   |   |   |--- pipeline-2__Race_WHITE <= 0.50
|   |   |   |   |   |   |   |   |   |--- pipeline-2__Education_Level_High School
Diploma <= 0.50
|   |   |   |   |   |   |   |   |   |   |--- class: False
|   |   |   |   |   |   |   |   |   |--- pipeline-2__Education_Level_High School
Diploma >  0.50
|   |   |   |   |   |   |   |   |   |   |--- class: False
|   |   |   |   |   |   |   |   |--- pipeline-2__Race_WHITE >  0.50
|   |   |   |   |   |   |   |   |   |--- pipeline-1__Jobs_Per_Year <= 1.02
```

```
|   |   |   |   |   |   |   |   |   |   |--- class: False
|   |   |   |   |   |   |   |   |   |--- pipeline-1__Jobs_Per_Year >  1.02
|   |   |   |   |   |   |   |   |   |   |--- class: False
|   |   |   |   |   |   |   |--- pipeline-1__Supervision_Risk_Score_First >
-0.67
|   |   |   |   |   |   |   |   |---
pipeline-2__Prior_Conviction_Episodes_Drug_1 <= 0.50
|   |   |   |   |   |   |   |   |   |--- pipeline-1__Jobs_Per_Year <= 0.28
|   |   |   |   |   |   |   |   |   |   |--- class: False
|   |   |   |   |   |   |   |   |   |--- pipeline-1__Jobs_Per_Year >  0.28
|   |   |   |   |   |   |   |   |   |   |--- class: False
|   |   |   |   |   |   |   |   |---
pipeline-2__Prior_Conviction_Episodes_Drug_1 >  0.50
|   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |--- pipeline-2__Prison_Offense_Violent/Sex >  0.50
|   |   |   |   |   |   |--- class: False
|   |   |--- pipeline-2__Gang_Affiliated_True >  0.50
|   |   |   |--- pipeline-2__Prior_Arrest_Episodes_Felony_1 <= 0.50
|   |   |   |   |--- pipeline-1__Jobs_Per_Year <= 0.24
|   |   |   |   |   |--- class: True
|   |   |   |   |--- pipeline-1__Jobs_Per_Year >  0.24
|   |   |   |   |   |--- pipeline-1__Jobs_Per_Year <= 0.30
|   |   |   |   |   |   |--- pipeline-1__DrugTests_THC_Positive <= 0.80
|   |   |   |   |   |   |   |--- pipeline-2__Prior_Conviction_Episodes_Drug_0 <=
0.50
|   |   |   |   |   |   |   |   |--- pipeline-1__Percent_Days_Employed <= 1.13
|   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |--- pipeline-1__Percent_Days_Employed >  1.13
|   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |--- pipeline-2__Prior_Conviction_Episodes_Drug_0 >
0.50
|   |   |   |   |   |   |   |   |--- pipeline-2__Prison_Offense_Property <= 0.50
|   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |--- pipeline-2__Prison_Offense_Property >  0.50
|   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |--- pipeline-1__DrugTests_THC_Positive >  0.80
|   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |--- pipeline-1__Jobs_Per_Year >  0.30
|   |   |   |   |   |   |--- pipeline-1__Jobs_Per_Year <= 1.36
|   |   |   |   |   |   |   |--- pipeline-1__Supervision_Risk_Score_First <=
0.60
|   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |--- pipeline-1__Supervision_Risk_Score_First >
0.60
|   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |--- pipeline-1__Jobs_Per_Year >  1.36
|   |   |   |   |   |   |   |--- pipeline-1__DrugTests_THC_Positive <= 1.03
|   |   |   |   |   |   |   |   |---
```

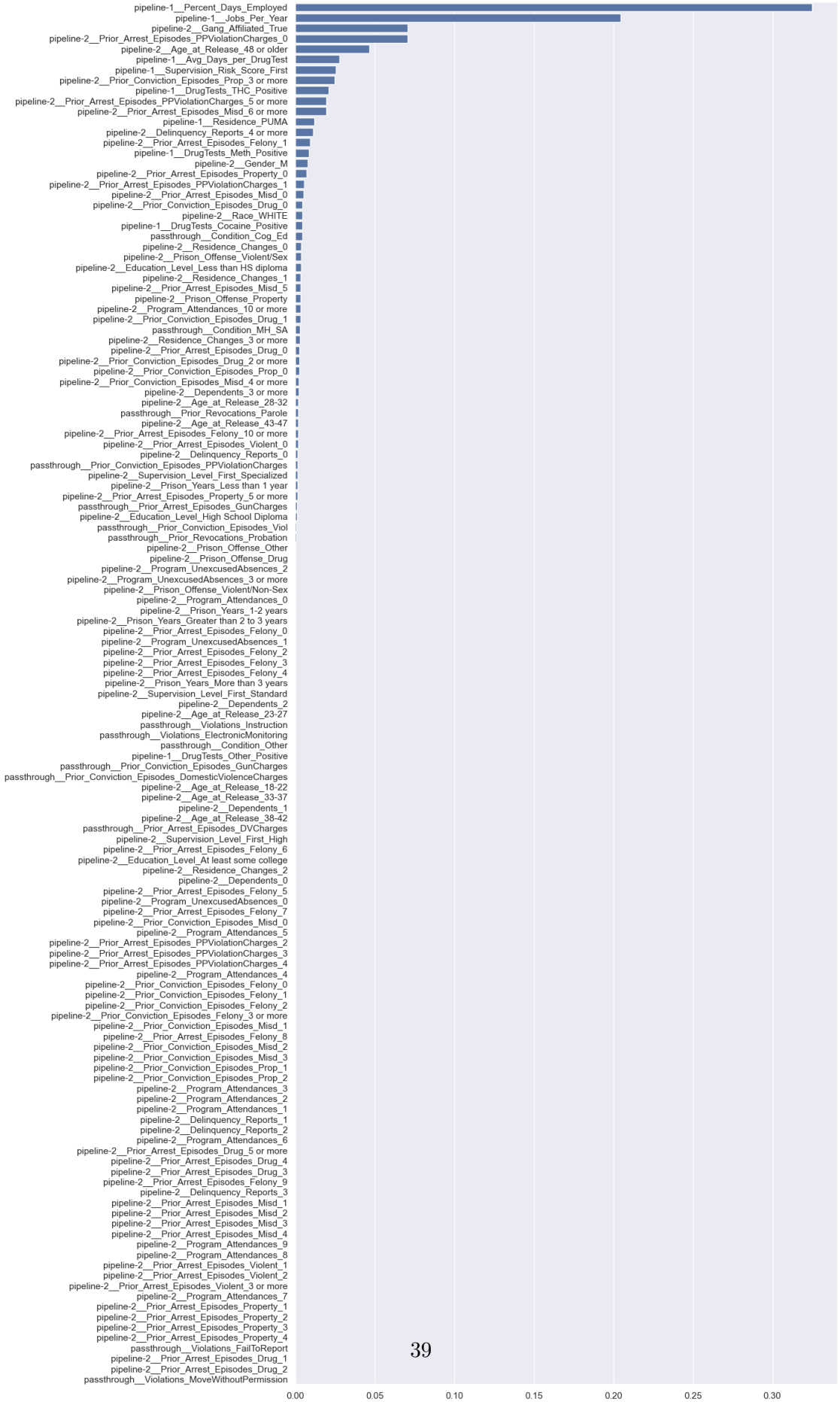
```
passthrough__Prior_Arrest_Episodes_GunCharges <= 0.50
|   |   |   |   |   |   |   |   |   |---
pipeline-2__Prior_Conviction_Episodes_Prop_0 <= 0.50
|   |   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |   |---
pipeline-2__Prior_Conviction_Episodes_Prop_0 >  0.50
|   |   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |   |---
passthrough__Prior_Arrest_Episodes_GunCharges >  0.50
|   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |---  pipeline-1__DrugTests_THC_Positive >  1.03
|   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |---  pipeline-2__Prior_Arrest_Episodes_Felony_1 >  0.50
|   |   |   |   |   |--- class: True
```

The method above was not satisfactory. In this case, visualizing all the rules of the decision tree as text is better. Because the tree structure is too complex and the tree size is relatively large in this case, image representation of the decision tree will not be ideal. Like the image above, it is difficult to see each node clearly from the image. Visualizing the decision tree rules as text provides a clearer picture of how the model makes decisions. The text representation lists the conditions for each node, making it easier to see the features and thresholds associated with each node.

When it is not possible to interpret a decision tree because of its complex structure, we can still extract other information from it that will help us understand the features' importance in the decision. The code in the cell below extracts the feature importances from the model (line 3), then uses this information to create a bar plot of features sorted by importance. The feature importance extracted this way is based on Gini Importance (as it is done in the original paper), which reflects how the features were picked when building the decision tree.

[ ]:
```python
import seaborn as sns

feature_importances = tree_model.named_steps["dt"].feature_importances_

# Sort the feature importances from greatest to least using the sorted indices
sorted_indices = feature_importances.argsort()[::-1]
sorted_feature_names = tree_model.named_steps['ct'].
  ↪get_feature_names_out()[sorted_indices]
sorted_importances = feature_importances[sorted_indices]

# # Create a bar plot of the feature importances
sns.set(rc={'figure.figsize':(11.7,30)})
sns.barplot(x=sorted_importances, y=sorted_feature_names)
```

[ ]:
```
<Axes: >
```


pipeline-1__Percent_Days_Employed
pipeline-1__Jobs_Per_Year
pipeline-2__Gang_Affiliated_True
pipeline-2__Prior_Arrest_Episodes_PPViolationCharges_0
pipeline-2__Age_at_Release_48 or older
pipeline-1__Avg_Days_per_DrugTest
pipeline-1__Supervision_Risk_Score_First
pipeline-2__Prior_Conviction_Episodes_Prop_3 or more
pipeline-1__DrugTests_THC_Positive
pipeline-2__Prior_Arrest_Episodes_PPViolationCharges_5 or more
pipeline-2__Prior_Arrest_Episodes_Misd_6 or more
pipeline-1__Residence_PUMA
pipeline-2__Delinquency_Reports_4 or more
pipeline-2__Prior_Arrest_Episodes_Felony_1
pipeline-1__DrugTests_Meth_Positive
pipeline-2__Gender_M
pipeline-2__Prior_Arrest_Episodes_Property_0
pipeline-2__Prior_Arrest_Episodes_PPViolationCharges_1
pipeline-2__Prior_Arrest_Episodes_Misd_0
pipeline-2__Prior_Conviction_Episodes_Drug_0
pipeline-2__Race_WHITE
pipeline-1__DrugTests_Cocaine_Positive
passthrough__Condition_Cog_Ed
pipeline-2__Residence_Changes_0
pipeline-2__Prison_Offense_Violent/Sex
pipeline-2__Education_Level_Less than HS diploma
pipeline-2__Residence_Changes_1
pipeline-2__Prior_Arrest_Episodes_Misd_5
pipeline-2__Prison_Offense_Property
pipeline-2__Program_Attendances_10 or more
pipeline-2__Prior_Conviction_Episodes_Drug_1
passthrough__Condition_MH_SA
pipeline-2__Residence_Changes_3 or more
pipeline-2__Prior_Arrest_Episodes_Drug_0
pipeline-2__Prior_Conviction_Episodes_Drug_2 or more
pipeline-2__Prior_Conviction_Episodes_Prop_0
pipeline-2__Prior_Conviction_Episodes_Misd_4 or more
pipeline-2__Dependents_3 or more
pipeline-2__Age_at_Release_28-32
passthrough__Prior_Revocations_Parole
pipeline-2__Age_at_Release_43-47
pipeline-2__Prior_Arrest_Episodes_Felony_10 or more
pipeline-2__Prior_Arrest_Episodes_Violent_0
pipeline-2__Delinquency_Reports_0
passthrough__Prior_Conviction_Episodes_PPViolationCharges
pipeline-2__Supervision_Level_First_Specialized
pipeline-2__Prison_Years_Less than 1 year
pipeline-2__Prior_Arrest_Episodes_Property_5 or more
passthrough__Prior_Arrest_Episodes_GunCharges
pipeline-2__Education_Level_High School Diploma
passthrough__Prior_Conviction_Episodes_Viol
passthrough__Prior_Revocations_Probation
pipeline-2__Prison_Offense_Other
pipeline-2__Prison_Offense_Drug
pipeline-2__Program_UnexcusedAbsences_2
pipeline-2__Program_UnexcusedAbsences_3 or more
pipeline-2__Prison_Offense_Violent/Non-Sex
pipeline-2__Program_Attendances_0
pipeline-2__Prison_Years_1-2 years
pipeline-2__Prison_Years_Greater than 2 to 3 years
pipeline-2__Prior_Arrest_Episodes_Felony_0
pipeline-2__Program_UnexcusedAbsences_1
pipeline-2__Prior_Arrest_Episodes_Felony_2
pipeline-2__Prior_Arrest_Episodes_Felony_3
pipeline-2__Prior_Arrest_Episodes_Felony_4
pipeline-2__Prison_Years_More than 3 years
pipeline-2__Supervision_Level_First_Standard
pipeline-2__Dependents_2
pipeline-2__Age_at_Release_23-27
passthrough__Violations_Instruction
passthrough__Violations_ElectronicMonitoring
passthrough__Condition_Other
pipeline-1__DrugTests_Other_Positive
passthrough__Prior_Conviction_Episodes_GunCharges
passthrough__Prior_Conviction_Episodes_DomesticViolenceCharges
pipeline-2__Age_at_Release_18-22
pipeline-2__Age_at_Release_33-37
pipeline-2__Dependents_1
pipeline-2__Age_at_Release_38-42
passthrough__Prior_Arrest_Episodes_DVCharges
pipeline-2__Supervision_Level_First_High
pipeline-2__Prior_Arrest_Episodes_Felony_6
pipeline-2__Education_Level_At least some college
pipeline-2__Residence_Changes_2
pipeline-2__Dependents_0
pipeline-2__Prior_Arrest_Episodes_Felony_5
pipeline-2__Program_UnexcusedAbsences_0
pipeline-2__Prior_Arrest_Episodes_Felony_7
pipeline-2__Prior_Conviction_Episodes_Misd_0
pipeline-2__Program_Attendances_5
pipeline-2__Prior_Arrest_Episodes_PPViolationCharges_2
pipeline-2__Prior_Arrest_Episodes_PPViolationCharges_3
pipeline-2__Prior_Arrest_Episodes_PPViolationCharges_4
pipeline-2__Program_Attendances_4
pipeline-2__Prior_Conviction_Episodes_Felony_0
pipeline-2__Prior_Conviction_Episodes_Felony_1
pipeline-2__Prior_Conviction_Episodes_Felony_2
pipeline-2__Prior_Conviction_Episodes_Felony_3 or more
pipeline-2__Prior_Conviction_Episodes_Misd_1
pipeline-2__Prior_Arrest_Episodes_Felony_8
pipeline-2__Prior_Conviction_Episodes_Misd_2
pipeline-2__Prior_Conviction_Episodes_Misd_3
pipeline-2__Prior_Conviction_Episodes_Prop_1
pipeline-2__Prior_Conviction_Episodes_Prop_2
pipeline-2__Program_Attendances_3
pipeline-2__Program_Attendances_2
pipeline-2__Program_Attendances_1
pipeline-2__Delinquency_Reports_1
pipeline-2__Delinquency_Reports_2
pipeline-2__Program_Attendances_6
pipeline-2__Prior_Arrest_Episodes_Drug_5 or more
pipeline-2__Prior_Arrest_Episodes_Drug_4
pipeline-2__Prior_Arrest_Episodes_Drug_3
pipeline-2__Prior_Arrest_Episodes_Felony_9
pipeline-2__Delinquency_Reports_3
pipeline-2__Prior_Arrest_Episodes_Misd_1
pipeline-2__Prior_Arrest_Episodes_Misd_2
pipeline-2__Prior_Arrest_Episodes_Misd_3
pipeline-2__Prior_Arrest_Episodes_Misd_4
pipeline-2__Program_Attendances_9
pipeline-2__Program_Attendances_8
pipeline-2__Prior_Arrest_Episodes_Violent_1
pipeline-2__Prior_Arrest_Episodes_Violent_2
pipeline-2__Prior_Arrest_Episodes_Violent_3 or more
pipeline-2__Program_Attendances_7
pipeline-2__Prior_Arrest_Episodes_Property_1
pipeline-2__Prior_Arrest_Episodes_Property_2
pipeline-2__Prior_Arrest_Episodes_Property_3
pipeline-2__Prior_Arrest_Episodes_Property_4
passthrough__Violations_FailToReport
pipeline-2__Prior_Arrest_Episodes_Drug_1
pipeline-2__Prior_Arrest_Episodes_Drug_2
passthrough__Violations_MoveWithoutPermission
0.00
0.05
0.10
0.15
0.20
0.25
0.30

Comment on the features importance of the tree model, compared to those seen in the logistic regression model, as well as the original paper results. Also, **what is a big limitation of using feature importance, compared to observing the coefficient of the logistic regression model?**

As mentioned above, the feature importance of the decision tree reflects how much it helps reduce uncertainty. Consistent with the findings of the original paper, decision trees excel in terms of accuracy, but are not inherently interpretable. For example, Percent_Days_Employed is one of the most important features in the tree model, but the decision tree is unable to explain in which direction the increase or decrease of these features is pushing the prediction. In logistic regression, by contrast, the positivity or negativity of the coefficients indicates the direction of the impact prediction, making it easier to interpret. A key limitation of tree-based feature significance is that there is no way to explain how each feature affects the prediction, whereas the coefficients of logistic regression provide more transparency and are easier to interpret.

5.0.5 Question 10

As before, we are interested in evaluating how the model classifies a particular sample. Let's start looking at the classification for our `hard_sample`. Is it correct?

[ ]:
```
# Your answer here
tree_model.predict(hard_sample)
```

[ ]:
```
array([ True])
```

For this hard sample, the prediction made by the decision tree isn't match the actual value. It is incorrect.

We would like to be able to tell what sequence of rules has led to this final decision, but, for a tree this large, this can be difficult, unless we want to manually sift through the list of rules or write some elaborate custom code. In the next sections, we will see an alternative method (SHAP) to achieve this result.

5.0.6 Question 11: Evaluation of Non-inherently Interpretable Models Using a Surrogate Model

Models that are not inherently interpretable ("black box" models) can still be examined to understand how they used the available features to make their predictions. In fact, there are many strategies to do this. The first one we are going to see is through use of a **surrogate model.** In this case, we train another model - an inherently interpretable one, such as a logistic regressor - on the *predictions* of the black box model, and then we try to interpret *its parameters*. Let's complete the code below to do that on the two non-inherently interpretable models included in this exercise: the Random Forest and XGBoost.

Surrogate for Random Forest Model

[ ]:
```
```

```
# Step 1: create logistic regressor object.
# For simplicity, we will use the already existing "NIJ_logreg.joblib" and␣
 ↪re-train it, instead of creating
# a new one. The reason for this decision is that NIJ_logreg.joblib already␣
 ↪knows how to handle the features
# of this dataset, while a new one will need to be designed to do so.

surrogate_model_rf = joblib.load("models_for_A3/NIJ_logreg.joblib")

# Step 2: train model on random forest predictions on the training set

rf_predictions = rf_model.predict(X_train)
surrogate_model_rf.fit(X_train, rf_predictions)

# Step 3: visualize weights of surrogate model, as we did for the original␣
 ↪logistic regression model

coefficients_rf = surrogate_model_rf.named_steps["logisticregression"].coef_.
 ↪flatten()
feature_names = np.array(surrogate_model_rf.named_steps['columntransformer'].
 ↪get_feature_names_out())

coeff_df_rf = pd.DataFrame({'Feature': feature_names, 'Coefficient':␣
 ↪coefficients_rf})

coeff_df_rf_sorted = coeff_df_rf.sort_values(by='Coefficient', ascending=False)

print("Top 10 positive coefficients:")
print(coeff_df_rf_sorted.head(10))

print("\nTop 10 negative coefficients:")
print(coeff_df_rf_sorted.tail(10))
```

```
Top 10 positive coefficients:
                                       Feature  Coefficient
11               pipeline-2__Age_at_Release_18-22     0.796636
18               pipeline-2__Gang_Affiliated_True     0.790565
95              pipeline-2__Delinquency_Reports_1     0.650981
38     pipeline-2__Prior_Arrest_Episodes_Felony_0     0.489534
12               pipeline-2__Age_at_Release_23-27     0.488824
9                            pipeline-2__Gender_M     0.446960
124         passthrough__Prior_Revocations_Parole     0.370422
126                  passthrough__Condition_MH_SA     0.357176
8                       pipeline-1__Jobs_Per_Year     0.319845
129  passthrough__Violations_ElectronicMonitoring     0.319349

Top 10 negative coefficients:
```

```
                                                Feature  Coefficient
97                    pipeline-2__Delinquency_Reports_3    -0.204958
72   pipeline-2__Prior_Arrest_Episodes_PPViolationC…    -0.230000
15                     pipeline-2__Age_at_Release_38-42    -0.238056
41           pipeline-2__Prior_Arrest_Episodes_Felony_2    -0.251144
16                     pipeline-2__Age_at_Release_43-47    -0.346385
101          pipeline-2__Program_Attendances_10 or more    -0.387002
98            pipeline-2__Delinquency_Reports_4 or more    -0.518417
39           pipeline-2__Prior_Arrest_Episodes_Felony_1    -0.547181
7                     pipeline-1__Percent_Days_Employed    -0.685253
17               pipeline-2__Age_at_Release_48 or older    -0.756514
```

Now that we have the weights of the surrogate model, what can we say about how the Random Forest model makes its predictions? What features seem more important? Are they similar to what we have seen for the other models so far?

Based on the surrogate model's weights, features with larger magnitudes seem more important in the Random Forest model's predictions. We can see that the Random Forest model has similar significant features to other models. Key features such as age at release, gang affiliation, and delinquency reports appear to play major roles in influencing the predictions. Features with positive coefficients, like younger age and gang affiliation, push the model's predictions towards recidivism. On the other hand, features with negative coefficients, such as older age and a higher number of delinquency reports, push predictions towards non-recidivism. Even the Random Forest modelis more complex, it still relies on similar important features to make predictions.

Note: using a surrogate model is not always a very good strategy, because the simpler "white box" model is often unable to replicate the behavior of the most complex "black box" model. We can get a sense of how close the surrogate is approximating the original model by looking at the R2 score. In the paper, they do so when trying to create a surrogate for XGBoost, and they explain:

The R2 value between the XGBoost predictions and the surrogate model predictions on the test set is 0.38. The surrogate model only explains 38% of the variance in the XGBoost model's predictions

Test this for the random forest surrogate model. How much variance is it able to capture?

Hints: - Think carefully about what constitues the array of predictions and the array of ground truths in this case - You may remember that R2 is, in fact, a metric for regression, not for classification! How can we use R2 in this case? There are various ways to approximate R2 for classification, as explained here. We will use the simplest one and use **count R2**, which is simply the accuracy of the surrogate classifier

[ ]:
```
# Your answer here
rf_predictions_train = rf_model.predict(X_train)
surrogate_predictions_rf_train = surrogate_model_rf.predict(X_train)

print("R^2 for random forest surrogate:", accuracy_score(rf_predictions_train,␣
 ↪surrogate_predictions_rf_train))
```

```
R^2 for random forest surrogate: 0.7180380446803805
```

The R^2 score is approximately 0.718 for the surrogate model, which means that the surrogate model successfully captures about 71.8% of the variance in the Random Forest model's predictions.

Overall, this R^2 is close to 1, which shows that the Random Forest's behavior is not too complex to approximate and can be interpreted to a significant degree through the surrogate model.

Now, repeat the analysis through surrogate model for XGBoost. Comment on the results, including considerations on the following: - What seem to be the most important features? - How do the sets of most important features compare across models (do not forget logistic regression and decision tree in this comparison)? - How good are the surrogate models, in terms of capturing the variance of the original model? Are they reliable? - ...more thoughts of your choice...

Surrogate for XGBoost Model

[ ]:
```
# Your answer here
surrogate_model_xgb = joblib.load("models_for_A3/NIJ_logreg.joblib")

xgb_predictions = xgboost_model.predict(X_train)
surrogate_model_xgb.fit(X_train, xgb_predictions)

coefficients_xgb = surrogate_model_xgb.named_steps["logisticregression"].coef_.
 ↪flatten()
feature_names = np.array(surrogate_model_rf.named_steps['columntransformer'].
 ↪get_feature_names_out())

coeff_df_xgb = pd.DataFrame({'Feature': feature_names, 'Coefficient':␣
 ↪coefficients_xgb})

coeff_df_xgb_sorted = coeff_df_xgb.sort_values(by='Coefficient',␣
 ↪ascending=False)

print("Top 10 positive coefficients:")
print(coeff_df_xgb_sorted.head(10))

print("\nTop 10 negative coefficients:")
print(coeff_df_xgb_sorted.tail(10))

xgb_predictions_train = xgboost_model.predict(X_train)
surrogate_predictions_xgb_train = surrogate_model_xgb.predict(X_train)

print("R^2 for XGBoost surrogate:", accuracy_score(xgb_predictions_train,␣
 ↪surrogate_predictions_xgb_train))
```

```
Top 10 positive coefficients:
                                          Feature  Coefficient
11                    pipeline-2__Age_at_Release_18-22     1.344978
18                    pipeline-2__Gang_Affiliated_True     1.322722
9                                pipeline-2__Gender_M     0.939579
95                   pipeline-2__Delinquency_Reports_1     0.853040
12                    pipeline-2__Age_at_Release_23-27     0.767391
38          pipeline-2__Prior_Arrest_Episodes_Felony_0     0.668436
126                       passthrough__Condition_MH_SA     0.643236
```

```
40   pipeline-2__Prior_Arrest_Episodes_Felony_10 or…     0.606695
36            pipeline-2__Prison_Years_Less than 1 year     0.591937
129       passthrough__Violations_ElectronicMonitoring     0.589146

Top 10 negative coefficients:
                                              Feature  Coefficient
35   pipeline-2__Prison_Years_Greater than 2 to 3 y…    -0.373178
41          pipeline-2__Prior_Arrest_Episodes_Felony_2    -0.378917
15                    pipeline-2__Age_at_Release_38-42    -0.383271
72   pipeline-2__Prior_Arrest_Episodes_PPViolationC…    -0.505892
16                    pipeline-2__Age_at_Release_43-47    -0.568909
101         pipeline-2__Program_Attendances_10 or more    -0.747395
98           pipeline-2__Delinquency_Reports_4 or more    -0.860045
39          pipeline-2__Prior_Arrest_Episodes_Felony_1    -0.914729
17              pipeline-2__Age_at_Release_48 or older    -1.321597
7                    pipeline-1__Percent_Days_Employed    -1.326048
R^2 for XGBoost surrogate: 0.8152510506525105
```

[ ]:
```
coeffs = logreg_model.named_steps["logisticregression"].coef_.flatten()
coeff_df = pd.DataFrame(coeffs, index=feature_names, columns=["Coefficient"])
coeff_df_sorted = coeff_df.sort_values(by="Coefficient", ascending=False)
coeff_df_sorted
```

[ ]:
```
                                          Coefficient
pipeline-2__Gang_Affiliated_True             0.777359
pipeline-2__Age_at_Release_18-22             0.769493
pipeline-2__Delinquency_Reports_1            0.635820
pipeline-2__Age_at_Release_23-27             0.488786
pipeline-2__Prior_Arrest_Episodes_Felony_0   0.473336
…                                                 …
pipeline-2__Program_Attendances_10 or more  -0.385807
pipeline-2__Prior_Arrest_Episodes_Felony_1  -0.501980
pipeline-2__Delinquency_Reports_4 or more   -0.507621
pipeline-1__Percent_Days_Employed           -0.663675
pipeline-2__Age_at_Release_48 or older      -0.752273

[133 rows x 1 columns]
```

From the surrogate model results, the top positive coefficients indicate the features that contribute the most towards predicting a positive class, include "Age_at_Release_18-22", "Gang_Affiliated_True", and "Gender_M". The top negative coefficients represent features that reduce the probability of a positive prediction. "Percent_Days_Employed" and "Age_at_Release_48 or older" are the most important negative features.

When we compare feature importance between XGBoost, logistic regression, and decision tree models, we can find some similar important features, such as "Gang_Affiliated" and "Age_at_Release" are important in all models. In particular, there is a lot of overlap between the important features of XGBoost and logistic regression, but the strength of the impact on the predictions is somewhat different. The decision tree will have some different important features present.

The R^2 score is approximately 0.815 for the surrogate model, which means that the surrogate model successfully captures about 81.5% of the variance in the XGBoost model.

Surrogate models are helpful to let the complex models more interpretable. In this case, the surrogate gives us a good idea of which features are most important. However, surrogate models simplify the original complex model, so it might missing and ignore some information, especially more complex relationships other than linear relationships between feature and feature.

5.0.7 Question 12: Evaluation of Non-inherently Interpretable Models Using Permutation Feature Importance

Another method used to interpret black box models is using feature permutation, which means changing the value of a feature and observing changes in the model's prediction error. More important features, when changed, will result in more frequent mistakes.

Luckily for us, Permutation Feature Importance already exists as a function in Scikit-Learn! All you have to do it is looking at the documentation to learn how it works, and apply it to the 3 non-inherently interpretable models of this exercise. Let's start with Random Forest.

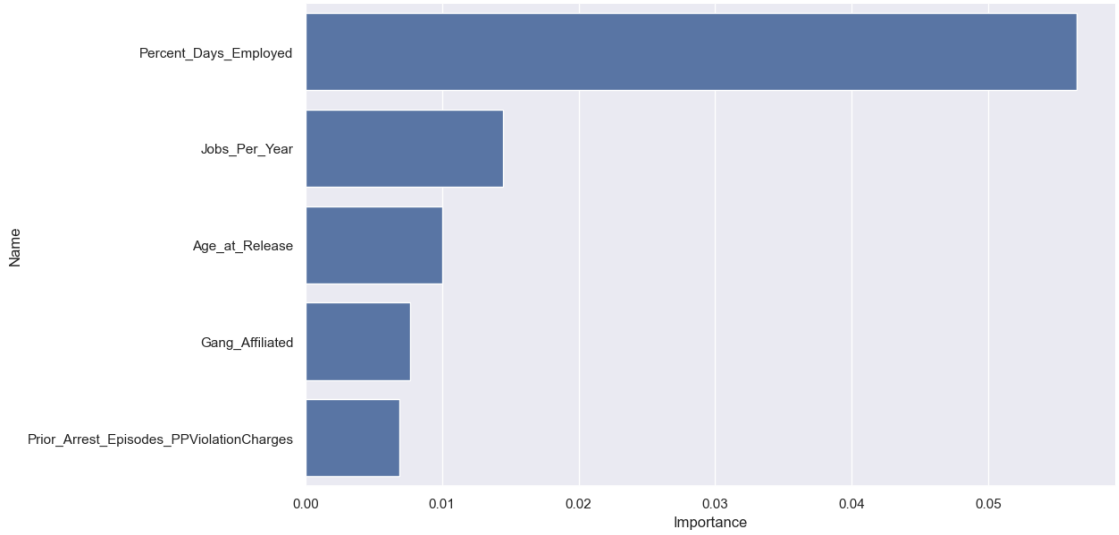
Random Forest Model:

[ ]:
```
# Use permutation_importance on the random forest model, and save the result in␣
↪a variable called "out"
out = permutation_importance(rf_model, X_test, y_test)
```

After you are done, you can run the cell below to visualize the top 5 most important features in a bar chart. If you like, you can change the number of features shown or try other visualization methods.

[ ]:
```
result = pd.DataFrame({"Name": X_test.columns, "Importance":␣
↪out["importances_mean"], "STD": out["importances_std"]})
result = result.sort_values(by=['Importance'], ascending=False)

sns.set(rc={'figure.figsize':(11.7,7)})
sns.barplot(data=result[:5], y="Name", x="Importance")
```

[ ]:
```
<Axes: xlabel='Importance', ylabel='Name'>
```



Now, use Permutation Feature Importance on XGBoost.

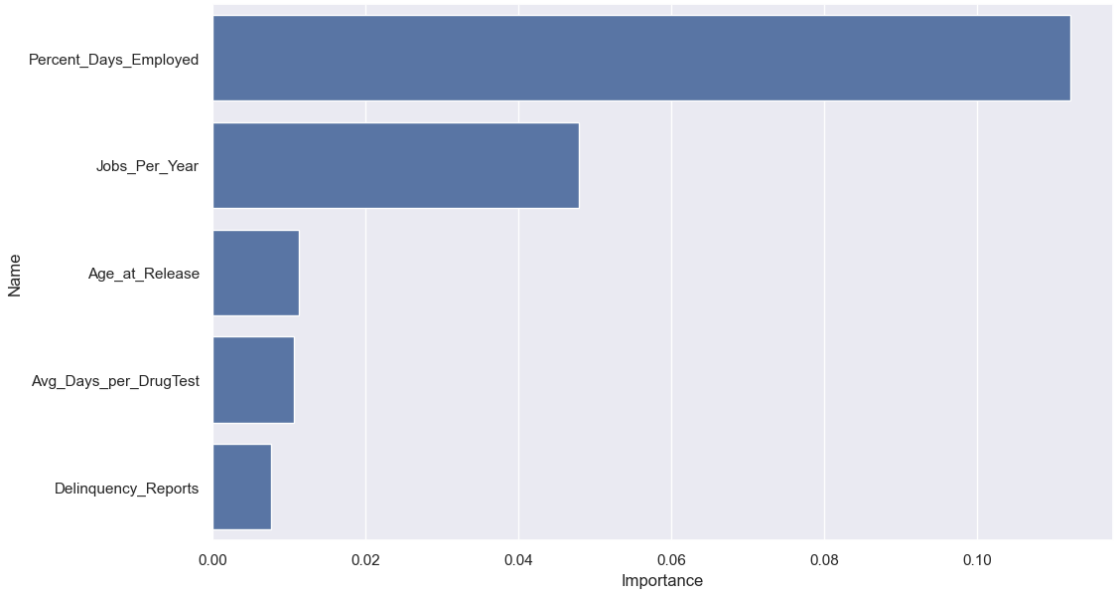
Hint: this is a more complex model; if you find that this task is taking too long, you may consider reducing the number of permutations using the parameter `n_repeats`. Be aware that this produces more variable results.

XGBoost Model:

[ ]:
```
out2 = permutation_importance(xgboost_model, X_test, y_test)
```

[ ]:
```
result2 = pd.DataFrame({"Name": X_test.columns, "Importance":␣
  ↪out2["importances_mean"], "STD": out2["importances_std"]})
result2 = result2.sort_values(by=['Importance'], ascending=False)

sns.set_theme(rc={'figure.figsize':(11.7,7)})
sns.barplot(data=result2[:5], y="Name", x="Importance")
```

[ ]:
```
<Axes: xlabel='Importance', ylabel='Name'>
```



Now that you have completed your analysis of feature importance using permutation, comment on the results. How do the sets of most important features compare with each other? Are this results similar to what you observed using the surrogate model?

Some important features, such as "Supervision_Risk_Score_First" and "Percent_Days_Employed," were determined to be the most significant after the permutation feature importance analysis was completed. This indicates that these features are crucial to the model's predictions because the accuracy of the model decreased the greatest when they were shuffled. By observing how the model's performance varies when a feature is altered, permutation importance helps demonstrate how important each feature is.

Both the surrogate model and permutaion can identify important features, such as "Gang_Affiliated" and "Age_at_Release_18-22." The surrogate model is simpler and more interperatable, but might ignore the complex interactions between features. The permutation method, could catch direct effects and interactions.

5.0.8 Question 13: Evaluation of Non-inherently Interpretable Models Using SHAP

The last method we are going to use to interpret the impact of each feature in our model is called SHAP, which stands for SHapley Additive exPlanations. How SHAP works is beyond the scope of this course, but if you are curious you can read the original paper by Lundberg and Lee and check out Lundberg's GitHub repo, which provides details on the implementation and examples.

You will need to install SHAP to be able to use it:

```
pip install shap
or
conda install -c conda-forge shap
```

Then, import it:

[ ]:
```
import shap # downgrade numpy to version = 1.23
shap.initjs()
```

```
<IPython.core.display.HTML object>
```

SHAP needs the model (we will start with Random Forest) and samples to use to explain the predictions. For this, we will need to give it transformed samples (scaled and imputed, as required by the model) from `X_train` or `X_test`.

[ ]:
```
X_train_enc = pd.DataFrame(
    data=rf_model.named_steps['ct'].transform(X_train),
    columns=feature_names,
    index=X_train.index,
)

X_test_enc = pd.DataFrame(
    data=rf_model.named_steps['ct'].transform(X_test),
    columns=feature_names,
    index=X_test.index,
)

ind = np.random.choice(len(X_test_enc) - 1, 1000)
# This line just gives 1000 random indexes from the training set
# We do this because getting SHAP values for all samples would be a bit too␣
 ↪long, but you
# are free to try it out!

ind = np.append(ind, 106) # adding the hard sample - we'll need this later
```

The following lines are all that's needed to explain the model's predictions for a set of samples:

[ ]:
```
rf_explainer = shap.Explainer(rf_model[-1])  # creating SHAP Explainer based on␣
 ↪the model

rf_shap_values = rf_explainer.shap_values(X_test_enc.iloc[ind])  # explaining␣
 ↪predictions for 1000 random samples
```

This gives us the SHAP values for each sample and each feature (the index 1 indicates the positive class):

[ ]:
```
rf_shap_values[1]
```

[ ]:
```
array([[ 4.49563052e-03, -4.49563052e-03],
       [-4.17405790e-02,  4.17405790e-02],
       [ 5.34912540e-03, -5.34912540e-03],
       [-4.64995947e-02,  4.64995947e-02],
       [ 2.55268469e-03, -2.55268469e-03],
       [ 7.10197236e-03, -7.10197236e-03],
       [ 3.93555675e-03, -3.93555675e-03],
```

```
[ 9.41014460e-02, -9.41014460e-02],
[ 5.94953910e-02, -5.94953910e-02],
[-2.76392666e-03,  2.76392666e-03],
[ 2.65000150e-03, -2.65000150e-03],
[ 2.79310520e-03, -2.79310520e-03],
[ 6.44508266e-03, -6.44508266e-03],
[-1.37122389e-02,  1.37122389e-02],
[-2.00559018e-03,  2.00559018e-03],
[-3.33935153e-04,  3.33935153e-04],
[-1.29773820e-03,  1.29773820e-03],
[-5.41176947e-03,  5.41176947e-03],
[ 1.51724320e-02, -1.51724320e-02],
[ 2.16823946e-03, -2.16823946e-03],
[-2.56072394e-03,  2.56072394e-03],
[-8.70889965e-03,  8.70889965e-03],
[-3.73377466e-03,  3.73377466e-03],
[-6.17770463e-03,  6.17770463e-03],
[-1.67705882e-03,  1.67705882e-03],
[ 4.04242769e-04, -4.04242769e-04],
[ 5.82729537e-04, -5.82729537e-04],
[-1.13697782e-03,  1.13697782e-03],
[-6.56219941e-04,  6.56219941e-04],
[ 4.42596631e-03, -4.42596631e-03],
[-4.04041431e-04,  4.04041431e-04],
[ 2.91488773e-03, -2.91488773e-03],
[ 1.93710766e-03, -1.93710766e-03],
[-4.37122125e-04,  4.37122125e-04],
[ 3.71873831e-03, -3.71873831e-03],
[-3.28076417e-03,  3.28076417e-03],
[ 1.11526298e-02, -1.11526298e-02],
[ 2.15527577e-02, -2.15527577e-02],
[ 6.23051726e-04, -6.23051726e-04],
[-2.88488873e-03,  2.88488873e-03],
[ 1.09516764e-02, -1.09516764e-02],
[-1.66541865e-03,  1.66541865e-03],
[-1.67883500e-03,  1.67883500e-03],
[-2.56473281e-04,  2.56473281e-04],
[-5.35217087e-03,  5.35217087e-03],
[ 3.26020705e-05, -3.26020705e-05],
[-3.08199260e-04,  3.08199260e-04],
[ 2.93272928e-04, -2.93272928e-04],
[ 9.89585282e-05, -9.89585282e-05],
[-2.11541651e-03,  2.11541651e-03],
[-1.04528690e-03,  1.04528690e-03],
[-1.41737324e-04,  1.41737324e-04],
[-6.66601299e-04,  6.66601299e-04],
[ 1.75188858e-04, -1.75188858e-04],
```

```
[-7.88484899e-05,  7.88484899e-05],
[-1.76802986e-02,  1.76802986e-02],
[-4.65754908e-03,  4.65754908e-03],
[ 5.77736817e-04, -5.77736817e-04],
[-7.76920791e-03,  7.76920791e-03],
[ 3.84724000e-04, -3.84724000e-04],
[ 1.31303793e-02, -1.31303793e-02],
[-3.17329094e-04,  3.17329094e-04],
[ 1.01050586e-04, -1.01050586e-04],
[ 2.02746796e-04, -2.02746796e-04],
[ 1.41245733e-04, -1.41245733e-04],
[ 4.12485583e-03, -4.12485583e-03],
[ 6.75621530e-04, -6.75621530e-04],
[ 7.19046504e-04, -7.19046504e-04],
[-3.44480440e-03,  3.44480440e-03],
[-9.42585839e-05,  9.42585839e-05],
[ 8.00022876e-04, -8.00022876e-04],
[-3.04033683e-03,  3.04033683e-03],
[-9.93710098e-03,  9.93710098e-03],
[-1.77321485e-05,  1.77321485e-05],
[-8.18704670e-04,  8.18704670e-04],
[ 3.36394186e-04, -3.36394186e-04],
[ 1.01652438e-03, -1.01652438e-03],
[-5.20179331e-02,  5.20179331e-02],
[-2.36025432e-03,  2.36025432e-03],
[-4.43921243e-03,  4.43921243e-03],
[ 3.02220554e-04, -3.02220554e-04],
[ 2.65771701e-03, -2.65771701e-03],
[-8.97492795e-03,  8.97492795e-03],
[ 1.72058366e-04, -1.72058366e-04],
[ 8.13327490e-04, -8.13327490e-04],
[ 5.23836232e-04, -5.23836232e-04],
[-4.01123224e-03,  4.01123224e-03],
[ 9.32975550e-03, -9.32975550e-03],
[ 1.38619390e-03, -1.38619390e-03],
[-2.74290311e-04,  2.74290311e-04],
[ 3.59605838e-03, -3.59605838e-03],
[ 5.73713216e-04, -5.73713216e-04],
[ 1.89683756e-04, -1.89683756e-04],
[-1.84057955e-03,  1.84057955e-03],
[ 3.12585604e-03, -3.12585604e-03],
[ 2.18419455e-03, -2.18419455e-03],
[ 3.71373727e-04, -3.71373727e-04],
[-5.51273366e-04,  5.51273366e-04],
[ 1.13221364e-02, -1.13221364e-02],
[-3.83328366e-03,  3.83328366e-03],
[ 3.85567671e-04, -3.85567671e-04],
```

```
       [-3.85685308e-03,  3.85685308e-03],
       [ 1.74979631e-04, -1.74979631e-04],
       [ 2.01921885e-04, -2.01921885e-04],
       [ 1.75978788e-04, -1.75978788e-04],
       [ 1.87808470e-04, -1.87808470e-04],
       [-7.88672386e-04,  7.88672386e-04],
       [ 2.97620244e-04, -2.97620244e-04],
       [-1.49812503e-04,  1.49812503e-04],
       [-3.64677940e-04,  3.64677940e-04],
       [ 6.79778802e-05, -6.79778802e-05],
       [ 7.48750524e-04, -7.48750524e-04],
       [-4.65133797e-05,  4.65133797e-05],
       [-1.27539692e-03,  1.27539692e-03],
       [ 1.62049187e-03, -1.62049187e-03],
       [ 1.50740838e-04, -1.50740838e-04],
       [-1.78977115e-04,  1.78977115e-04],
       [ 1.25065976e-03, -1.25065976e-03],
       [-8.93462753e-03,  8.93462753e-03],
       [-6.56855429e-04,  6.56855429e-04],
       [ 1.99682756e-03, -1.99682756e-03],
       [ 1.93106525e-04, -1.93106525e-04],
       [ 1.71493384e-03, -1.71493384e-03],
       [-2.93765364e-04,  2.93765364e-04],
       [ 1.69713605e-03, -1.69713605e-03],
       [-4.48160854e-04,  4.48160854e-04],
       [-9.94004879e-03,  9.94004879e-03],
       [ 2.02084539e-03, -2.02084539e-03],
       [ 1.98208019e-03, -1.98208019e-03],
       [ 1.12504190e-03, -1.12504190e-03],
       [-6.40912666e-03,  6.40912666e-03],
       [-1.96518926e-04,  1.96518926e-04],
       [-1.59197929e-03,  1.59197929e-03]])
```

This is hardly interpretable, though. It is better to get the average values for each feature, which returns something similar to feature importance:

[ ]:
```
values = np.abs(rf_shap_values[:,:,1]).mean(0)
pd.DataFrame(data=values, index=feature_names, columns=["SHAP"]).sort_values(
    by="SHAP", ascending=False)[:10]
```

[ ]:

	SHAP
pipeline-1__Percent_Days_Employed	0.072999
pipeline-2__Gang_Affiliated_True	0.027638
pipeline-1__Supervision_Risk_Score_First	0.025441
pipeline-1__DrugTests_THC_Positive	0.023567
pipeline-1__Jobs_Per_Year	0.021538
pipeline-2__Prior_Arrest_Episodes_PPViolationCh…	0.020443
pipeline-2__Prior_Arrest_Episodes_PPViolationCh…	0.016987

```
pipeline-2__Age_at_Release_48 or older              0.014727
pipeline-2__Prior_Conviction_Episodes_Misd_0        0.013314
pipeline-1__DrugTests_Meth_Positive                 0.012513
```

The SHAP library also has a lot of ways to visualize and interpret the SHAP values - try it out!

[ ]:
```
shap_figure = shap.summary_plot(rf_shap_values[:, :, 1], X_test_enc.iloc[ind],␣
↪plot_size=[12,6])
```

Given the new information obtained using the SHAP library on the Random Forest model, explain the results (you will need to refer to the SHAP documentation - or ask us for help interpreting the plots) and comment on the difference between these results and those obtained using the other methods.

Percent_Days_Employed has the largest impact on the model output compared to other features. Specifically, when the percent of days employed is lower, it pushes the prediction toward the positive class, indicating higher risk, and vice versa.

A high value of **Gang_Affiliated_True8** strongly pushes the prediction upward, implying that gang affiliation significantly increases the likelihood of the positive class.

Supervision_Risk_Score_First, DrugTests_THC_Positive, and Jobs_Per_Year also have a considerable impact. Higher scores for these features positively affect the model output, increasing the risk prediction. However, while DrugTests_THC_Positive has a larger positive impact, the other two features tend to have more negative impacts.

Age_at_Release_48 or older shows a negative impact on the model, with older individuals being less likely to be labeled as at risk.

According to the plot, Percent_Days_Employed, Gang_Affiliated_True, Age_at_Release_48 or older, and Jobs_Per_Year are important features. This aligns closely with the feature importance

from other methods. However, Delinquency_Reports_1 and Delinquency_Reports_4 or more are not considered important here.

Next, **repeat this analysis for XGBoost.**

[ ]:
```
# Your answer here

xgboost_explainer = shap.Explainer(xgboost_model[-1])  # creating SHAP␣
 ↪Explainer based on the model

xgboost_shap_values = xgboost_explainer.shap_values(X_test_enc.iloc[ind])  #␣
 ↪explaining predictions for 1000 random samples
```

[ ]:
```
values_xgboost = np.abs(xgboost_shap_values[1]).mean(0)
pd.DataFrame(data=values_xgboost, index=feature_names, columns=["SHAP"]).
 ↪sort_values(
    by="SHAP", ascending=False
)[:10]
```

[ ]:

	SHAP
pipeline-1__Residence_PUMA	0.042684
pipeline-2__Prior_Conviction_Episodes_Misd_2	0.042684
pipeline-2__Delinquency_Reports_4 or more	0.042684
pipeline-2__Delinquency_Reports_3	0.042684
pipeline-2__Delinquency_Reports_2	0.042684
pipeline-2__Delinquency_Reports_1	0.042684
pipeline-2__Delinquency_Reports_0	0.042684
pipeline-2__Prior_Conviction_Episodes_Drug_2 or…	0.042684
pipeline-2__Prior_Conviction_Episodes_Drug_1	0.042684
pipeline-2__Prior_Conviction_Episodes_Drug_0	0.042684

[ ]:
```
xgboost_figure = shap.summary_plot(xgboost_shap_values, X_test_enc.iloc[ind],␣
 ↪plot_size=[12,6])
```

Percent_Days_Employed has the strongest negative impact, with higher employment pushing the prediction downward toward the negative class. If Percent_Days_Employed is high, the individual is more likely to be predicted as belonging to the negative class. In contrast, Jobs_Per_Year and Gang_Affiliated_True both have a large positive impact on the model's output when their values are high.

According to the plot, Percent_Days_Employed, Gang_Affiliated_True, Age_at_Release_48 or older, drugtest_meth_positive and Jobs_Per_Year are important features. This aligns closely with the feature importance from other methods. However, Delinquency_Reports_1 and Delinquency_Reports_4 or more are not considered important here.

5.0.9 Question 14: Explaining individual predictions using SHAP

Another powerful feature of SHAP is that it allows us to explain the impact of each feature on individual predictions. For example, we will be able to explain how the prediction for our hard sample was generated. Let's start by looking at the prediction for this sample given by the random forest model. **Is it correct?**

Yes, correct.

[ ]:
```
shap_values = rf_explainer.shap_values(X_test_enc.iloc[ind[-1]])

pd.DataFrame(
    shap_values[:,1],
    index=feature_names,
    columns=["SHAP values"]).sort_values(
    by="SHAP values", ascending=False)
```

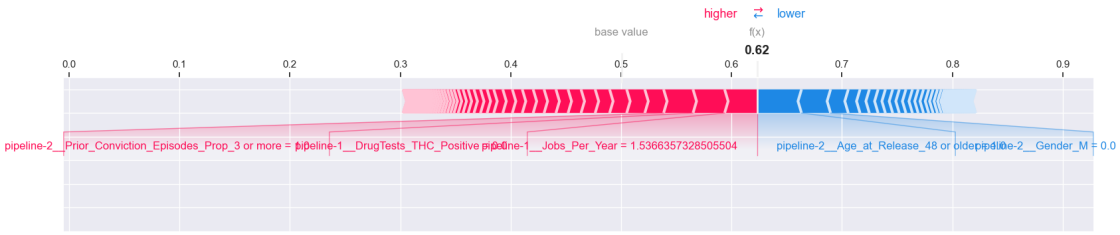
[ ]:

	SHAP values
pipeline-1__Jobs_Per_Year	0.029339

```
pipeline-1__DrugTests_THC_Positive                      0.028623
pipeline-2__Prior_Conviction_Episodes_Prop_3 or…     0.026641
pipeline-2__Prior_Arrest_Episodes_Property_5 or…     0.017039
pipeline-2__Prior_Conviction_Episodes_Prop_0            0.016405
…                                                          …
pipeline-2__Prison_Years_Less than 1 year             -0.008464
pipeline-2__Prior_Arrest_Episodes_PPViolationCh…    -0.010623
pipeline-2__Gang_Affiliated_True                       -0.016029
pipeline-2__Gender_M                                   -0.026349
pipeline-2__Age_at_Release_48 or older                 -0.040279

[133 rows x 1 columns]
```

Let's look at the **force plot** for this particular prediction, by running the cell below:

[ ]:
```
shap.force_plot(
    rf_explainer.expected_value[1],
    shap_values[:,1],
    X_test_enc.iloc[ind[-1]],
    matplotlib=True,
)
```



Interpret the plot results,, including the following: - What contributed the most to the prediction? - What countered the prediction the most? - Can we tell, by looking at the plot, that this was a difficult prediction?

1. Jobs_Per_Year contributed the most. The larger the SHAP value of a feature, the greater the positive impact on the model's output. This feature has the largest positive SHAP value (around 0.0293), significantly pushing the model's prediction toward the positive class. For this specific individual, he or she did 1.5 jobs per year, making them more likely to be labeled as at risk.

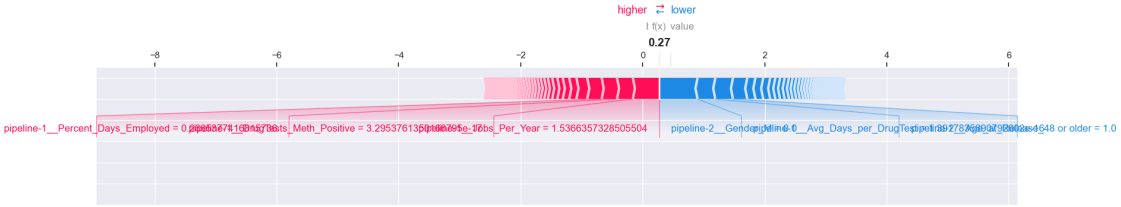
2. Age_at_Release_48 or older countered the most to the prediction. It has the largest negative SHAP value (around -0.040), indicating that it had the greatest negative effect on the model's output, pulling the prediction toward a lower value. This individual was released at 48 years or older, which makes him or her less likely to be labeled as at risk.

3. Yes, this was a difficult prediction. The relatively balanced contributions of positive and negative features made it challenging. The expected value of the prediction is around 0.63,

which is only slightly higher than the base value (0.5), indicating that the model was not strongly confident in its decision.

Finally, **repeat the analysis and comment on the results of the individual predictions made on the hard sample by XGBoost and Decision Tree** (since we were not able to do the latter earlier).

[ ]:
```
# Your answer here

shap.force_plot(
    xgboost_explainer.expected_value,
    xgboost_shap_values[-1],
    X_test_enc.iloc[ind[-1]],
    matplotlib=True,
)
```



Jobs_Per_Year contributed the most. This feature has the largest positive SHAP value (around 0.4309), significantly pushing the model's prediction toward the positive class. For this individual, having 1.54 jobs per year makes them more likely to be labeled as at risk.

Gender_M countered the prediction the most. It has the largest negative SHAP value (around -0.6299), indicating that it had the greatest negative impact, pulling the prediction toward a lower value. Since this individual is male, the model was less likely to predict them as at risk.

No, this was not a challenging prediction. The predicted value is 0.27, which is slightly far from the baseline (0.5). Therefore, we can reasonably predict that this individual has a low risk.

[ ]:
```
tree_explainer = shap.Explainer(tree_model[-1])  # creating SHAP Explainer␣
↪based on the model

tree_shap_values = tree_explainer.shap_values(X_test_enc.iloc[ind])
```

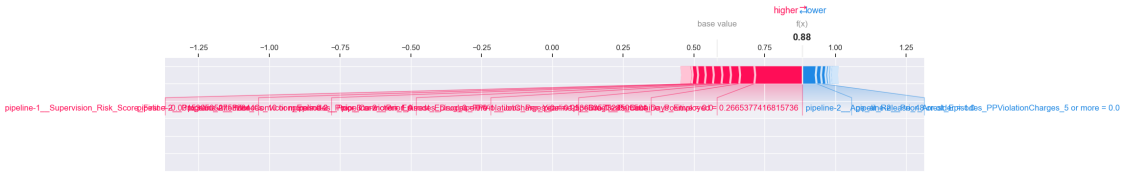
[ ]:
```
pd.DataFrame(
    tree_shap_values[-1,:,-1],
    index=feature_names,
    columns=["SHAP values"]).sort_values(
    by="SHAP values", ascending=False)
```

[ ]:

	SHAP values
pipeline-1__Percent_Days_Employed	0.168616

```
pipeline-1__DrugTests_Cocaine_Positive                  0.053310
pipeline-1__Jobs_Per_Year                               0.032276
pipeline-2__Prior_Arrest_Episodes_PPViolationCh…        0.030374
pipeline-2__Prior_Conviction_Episodes_Drug_0            0.030371
…                                                            …
pipeline-2__Prior_Arrest_Episodes_Misd_6 or more       -0.008180
pipeline-2__Prior_Arrest_Episodes_PPViolationCh…       -0.009242
pipeline-2__Gang_Affiliated_True                       -0.015517
pipeline-2__Prior_Arrest_Episodes_PPViolationCh…       -0.018721
pipeline-2__Age_at_Release_48 or older                 -0.051427

[133 rows x 1 columns]
```

[ ]:
```
shap.force_plot(
    tree_explainer.expected_value[1],
    tree_shap_values[-1,:,-1],
    X_test_enc.iloc[ind[-1]],
    matplotlib=True,
)
```



Supervision_Risk_Score_First contributed the most. It has the largest positive SHAP value (around 1.25), strongly pushing the prediction toward the positive class. This suggests that a high supervision risk score indicates a greater likelihood of being labeled as at risk.

Percent_Days_Employed countered the prediction the most. It has the largest negative SHAP value (around -0.88), indicating that higher employment pushes the prediction downward toward the negative class. This suggests that the more days employed, the less likely the individual is to be considered at risk.

No, this was not a difficult prediction. The final predicted value is 0.88, which is far from the baseline value (0.5). We can confidently predict that this individual belongs to the positive class.

6 Part 5: Final Evaluation:

6.0.1 Question 15

Using **all the results collected so far** on accuracy, fairness and transparency of the 5 models, write your recommendation about what model, in your opinion, should be employed for this application (300 words max).

From the perspective of model fairness and accuracy, we would prefer to use the XGBoost model

to predictwhether or not the person has committed a new felony or misdemeanour within 3 years from the beginning of parole supervision. The XGBoost model has the most balanced performance on the fairness metrics. This suggests that this model is more fair and treats the predictions of different populations more equitably. Also XGBoost did well in accuracy, its predictions have the smallest average distance from the reference value. It has to be recognized that XGBoost also has limitations as it is a black box model with limited interpretability. Nonetheless, it provides us with fairer and more accurate prediction results, which compensates for the lack of interpretability to some extent. In subsequent research, if one wants to increase the interpretability of this model, perhaps one can try to use a surrogate model, which helps to visualize the importance of different features in doing the prediction process.

7 Final thoughts

1) If you have completed this assignment in a group, please write a detailed description of how you divided the work and how you helped each other completing it:

We discussed our thoughts on the questions and shared our ideas with each other. And we divide the work so that each person could focus on specific parts of the assignment. We used GitHub to share our progress and make sure everything stayed organized. Once we completed all the questions, we came together to review the answers as a group, making sure that everything was clear and correct.

2) Have you used ChatGPT or a similar Large Language Model (LLM) to complete this homework? Please describe how you used the tool. We will never deduct points for using LLMs for completing homework assignments, but this helps us understand how you are using the tool and advise you in case we believe you are using it incorrectly.

We used ChatGPT to help us analyze and troubleshoot errors in our code when we couldn't resolve them on our own. ChatGPT provided suggestions on where the issues might be and how to fix them. Additionally, for this assignment, which involved a lot of written explanations, we used ChatGPT to help correct grammar mistakes in some parts of our written responses.

3) Have you struggled with some parts (or all) of this homework? Do you have pending questions you would like to ask? Write them down here!

We're unsure if we fully grasped how to explain the similarities and differences between the surrogate model and permutaion.