Module 3 - Algorithm auditing: Accuracy, Fairness and Interpretability

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.

Group members

Leave blanks if group has less than 2 members:

- Student 1: Jingyuan Liu (S.N. 69763183)
- Student 2: Nicholas Tam (S.N. 45695970)

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

Import Libraries:

```
# Here are some libraries you may need for this exercise, for your
convenience
#!pip install scikit-learn==1.1.0
import matplotlib.pyplot as plt
```

```
import numpy as np
import pandas as pd
#import seaborn as sns
# !pip install xgboost
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, # Depreciated, use ConfusionMatrixDisplay
    ConfusionMatrixDisplay,
    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


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

# !pip install eli5
import eli5

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

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")
```

```
[15:51:12] WARNING: /Users/runner/miniforge3/conda-bld/xgboost-
split_1667414697403/work/src/learner.cc:553:
  If you are loading a serialized model (like pickle in Python, RDS in
R) generated by
  older XGBoost, please export the model by calling
`Booster.save_model` from that version
  first, then load it back in current version. See:
```

```
https://xgboost.readthedocs.io/en/latest/tutorials/saving_model.html

  for more details about differences between saving model and
serializing.
```

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).**

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.

Add remaining definitions and formulas here

- **Accuracy:** A measure of how often classifier correctly predicts, as a proportion of all predicted instances that are correct between 0 and 1. A higher accuracy indicates that the model's predictions align well with the actual labels for each instance. While this measure is relatively easy to explain and allows for sensitivity for false predictions, it operates under the assumption that the classes in the population are balanced, and could be misleading if they are not; notably, it cannot express any uncertainty about imbalanced predictions.
  - Formula: $\frac{TP+TN}{TP+FP+FN+TN}$
- **Precision:** The proportion of true positives out of all positive predictions, measured between 0 and 1. A higher precision indicates the model is less likely to provide false positives. By measuring the model correctness in identifying the target class, it is effective for mitigating issues involving imbalanced classes, and allows for a more direct

interpretation for testing false positives (e.g. Minimise risk of email being incorrectly labeled as spam), though it does not account for false negatives in the data.

  - Formula: $\frac{TP}{TP+FP}$

- **Recall:** The proportion of true positives out of all actual positive instances, measured between 0 and 1. By measuring the model's ability to find objects of the target class, it is effective for mitigating issues involving imbalanced classes, and allows for a more direct interpretation for testing false negatives (e.g. Minimise risk of missing disease), though it does not account for false positives in the data.
  - Formula: $\frac{TP}{TP+FN}$
- **F1 score:** A measure of the harmonic mean of precision and recall. It is effective for mitigating issues involving imbalanced classes, and allows for the model to be tested for high and balanced values of precision and recall. However, it operates under the assumption that both precision and recall have equal weighting in the issue at hand, and does not account for the distribution of errors. It is also generally difficult to interpret, and is less effective with multi-class classification.
  - Formula: $\frac{2*Precision*Recall}{Precision+Recall}=\frac{2TP}{2TP+FP+FN}$
- Sources
  - Evidently AI Team. (n.d.). Accuracy vs. precision vs. recall in machine learning: What’s the difference? Evidently AI - Open-Source ML Monitoring and Observability. https://www.evidentlyai.com/classification-metrics/accuracy-precision-recall#:~:text=Cons%3A,performance%20on%20the%20target%20class.
  - The importance of accuracy in machine learning: A comprehensive guide. The Importance of Accuracy in Machine Learning: A Comprehensive Guide. (n.d.). https://www.artsyltech.com/blog/Accuracy-In-Machine-Learning
  - Kumar, S. (2024, October 8). Metrics to evaluate your classification model to take the right decisions. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/07/metrics-to-evaluate-your-classification-model-to-take-the-right-decisions/
  - Machine Learning. (2023, September 27). What are the advantages and disadvantages of using F1 score for Ann Performance Evaluation?. F1 Score for ANN Performance Evaluation: Pros and Cons. https://www.linkedin.com/advice/3/what-advantages-disadvantages-using-f1-score-ann

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.

LogReg Model:

```
# Compute required metrics here. You may add more cells if needed
data = {
    "model & dataset": [],
    "accuracy": [],
    "precision": [],
    "recall": [],
    "f1 score": [],
    "roc_auc_score": [],
}

def classifier_metrics(modelname, model):
    data["model & dataset"].append(str(modelname) + " " +
str("(Training)"))
    data["accuracy"].append(accuracy_score(y_train,
model.predict(X_train)))
    data["precision"].append(precision_score(y_train,
model.predict(X_train), zero_division=1))
    data["recall"].append(recall_score(y_train,
model.predict(X_train)))
    data["f1 score"].append(f1_score(y_train, model.predict(X_train)))
    data["roc_auc_score"].append(roc_auc_score(y_train,
model.predict(X_train)))

    data["model & dataset"].append(str(modelname) + " " +
str("(Testing)"))
    data["accuracy"].append(accuracy_score(y_test,
model.predict(X_test)))
    data["precision"].append(precision_score(y_test,
model.predict(X_test)))
    data["recall"].append(recall_score(y_test, model.predict(X_test)))
    data["f1 score"].append(f1_score(y_test, model.predict(X_test)))
    data["roc_auc_score"].append(roc_auc_score(y_test,
model.predict(X_test)))

data = {
    "model & dataset": [],
    "accuracy": [],
    "precision": [],
    "recall": [],
    "f1 score": [],
    "roc_auc_score": [],
}
classifier_metrics("logreg_model", logreg_model)
```

```
df = pd.DataFrame(data)
df

             model & dataset  accuracy  precision    recall  f1 score  \
0  logreg_model (Training)  0.714610   0.776119  0.716321  0.745022   
1   logreg_model (Testing)  0.701974   0.749397  0.709685  0.729000   

   roc_auc_score  
0       0.714273  
1       0.700825  
```

Random Forest Model:

```
# Compute required metrics here. You may add more cells if needed
data = {
    "model & dataset": [],
    "accuracy": [],
    "precision": [],
    "recall": [],
    "f1 score": [],
    "roc_auc_score": [],
}
classifier_metrics("rf_model", rf_model)
df = pd.DataFrame(data)
df

         model & dataset  accuracy  precision    recall  f1 score
roc_auc_score
0  rf_model (Training)  0.996627   0.999332  0.994870  0.997096
0.996972
1   rf_model (Testing)  0.719262   0.721262  0.819781  0.767372
0.704287
```

Decision Tree Model:

```
# Compute required metrics here. You may add more cells if needed
data = {
    "model & dataset": [],
    "accuracy": [],
    "precision": [],
    "recall": [],
    "f1 score": [],
    "roc_auc_score": [],
}
classifier_metrics("tree_model", tree_model)
df = pd.DataFrame(data)
df

           model & dataset  accuracy  precision    recall  f1 score  \
0  tree_model (Training)  0.738609   0.761806  0.801539  0.781168   
```

```
1   tree_model (Testing)  0.697845   0.713596  0.776839  0.743876

   roc_auc_score
0       0.726252
1       0.686077
```

XGBoost Model:

```
# Compute required metrics here. You may add more cells if needed
data = {
    "model & dataset": [],
    "accuracy": [],
    "precision": [],
    "recall": [],
    "f1 score": [],
    "roc_auc_score": [],
}
classifier_metrics("xgboost_model", xgboost_model)
df = pd.DataFrame(data)
df
```

```
            model & dataset  accuracy  precision    recall  f1
score  \
0  xgboost_model (Training)  0.873590   0.867201  0.924378  0.894877

1   xgboost_model (Testing)  0.735905   0.737228  0.827318  0.779679


   roc_auc_score
0       0.863618
1       0.722287
```

Overall:

```
data = {
    "model & dataset": [],
    "accuracy": [],
    "precision": [],
    "recall": [],
    "f1 score": [],
    "roc_auc_score": [],
}
# modellist = ["logreg_model", "rf_model", "tree_model",
"xgboost_model"]
# modeldict = {"logreg_model": logreg_model, "rf_model": rf_model,
"tree_model": tree_model, "xgboost_model": xgboost_model}

models = [logreg_model, rf_model, tree_model, xgboost_model]
model_names = ["LogReg Model", "Random Forest Model", "Decision Tree
Model ", "XGBoost Model"]
```

```
for model, model_name in zip(models,model_names):
# for model in modellist:
    classifier_metrics(model_name, model)
df = pd.DataFrame(data)
# df = df.set_index(["model & dataset"])
df
```

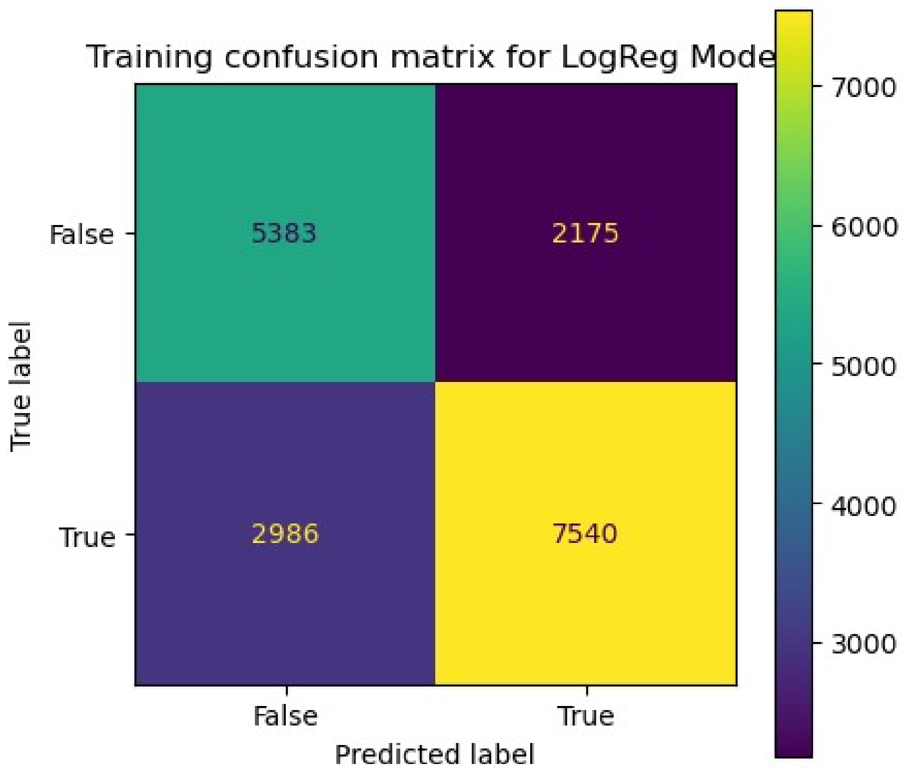
```
                   model & dataset  accuracy  precision    recall   f1
score  \
0         LogReg Model (Training)  0.714610   0.776119  0.716321
0.745022
1          LogReg Model (Testing)  0.701974   0.749397  0.709685
0.729000
2  Random Forest Model (Training)  0.996627   0.999332  0.994870
0.997096
3   Random Forest Model (Testing)  0.719262   0.721262  0.819781
0.767372
4  Decision Tree Model  (Training)  0.738609   0.761806  0.801539
0.781168
5   Decision Tree Model  (Testing)  0.697845   0.713596  0.776839
0.743876
6        XGBoost Model (Training)  0.873590   0.867201  0.924378
0.894877
7         XGBoost Model (Testing)  0.735905   0.737228  0.827318
0.779679

   roc_auc_score
0       0.714273
1       0.700825
2       0.996972
3       0.704287
4       0.726252
5       0.686077
6       0.863618
7       0.722287
```

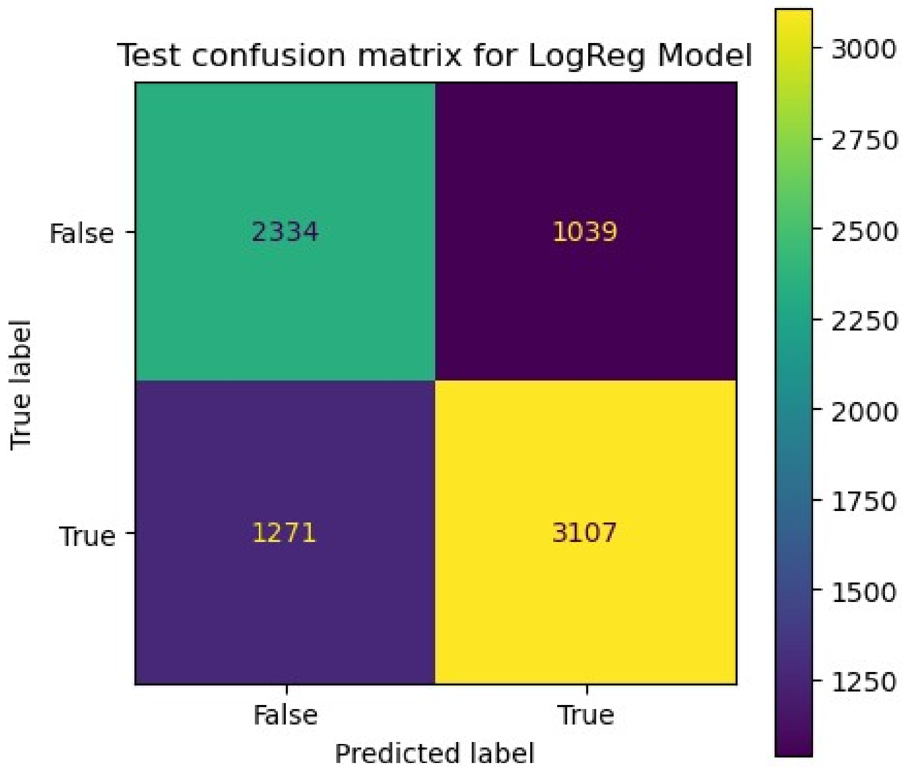
Question 3

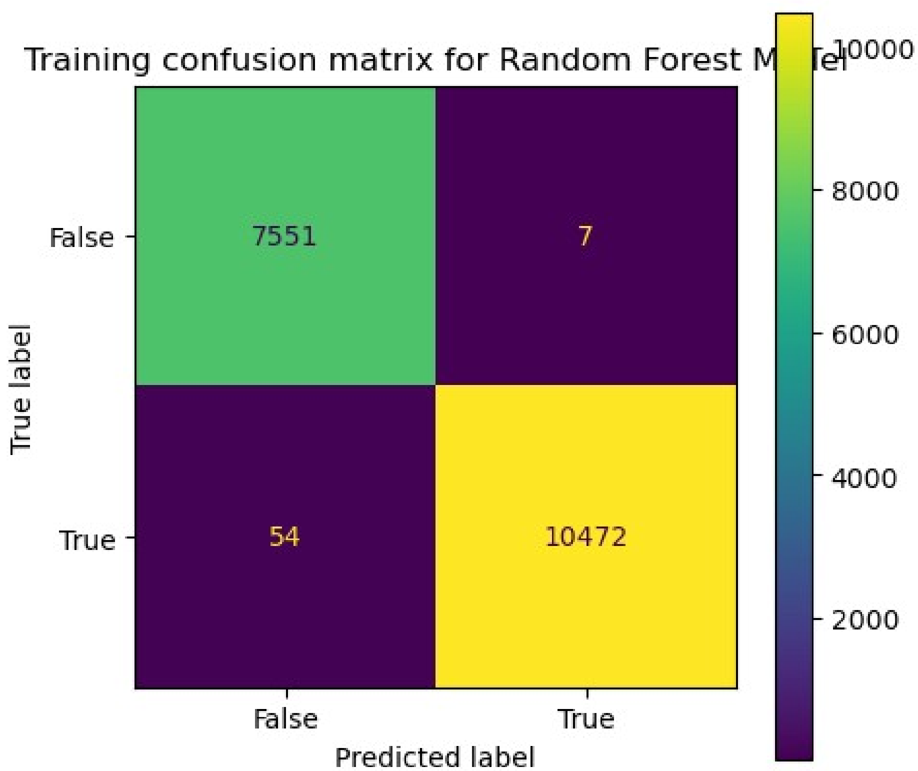
For every classifier given, plot the confusion matrices on training and test set. Here is another function you will find helpful for this task: `confusion_matrix`.

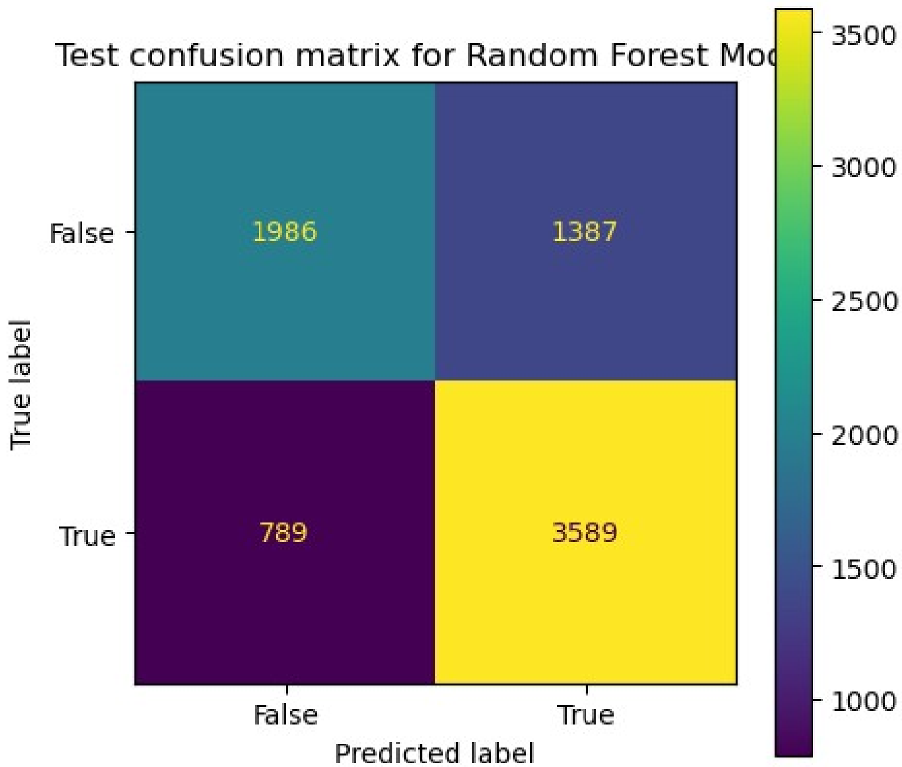
```
# Output confusion matrices here. You may add more cells if needed

# modellist = ["logreg_model", "rf_model", "tree_model",
"xgboost_model"]
# modeldict = {"logreg_model": logreg_model, "rf_model": rf_model,
"tree_model": tree_model, "xgboost_model": xgboost_model}

models = [logreg_model, rf_model, tree_model, xgboost_model]
```

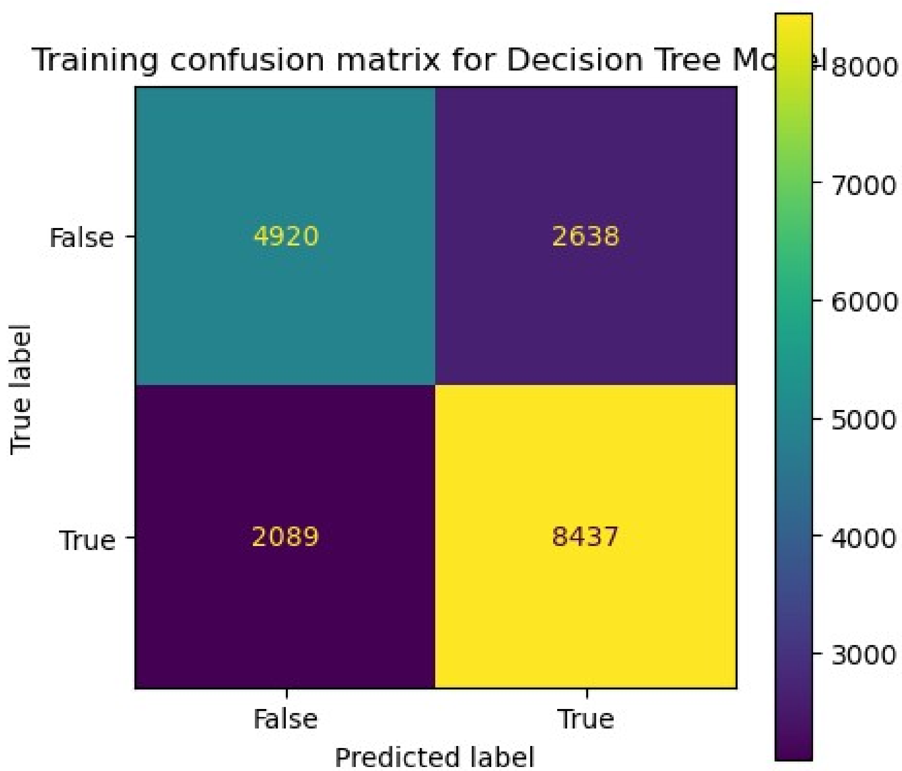
```
model_names = ["LogReg Model", "Random Forest Model", "Decision Tree
Model ", "XGBoost Model"]
for model, model_name in zip(models,model_names):
# for modelname in modellist:
    fig, ax = plt.subplots(figsize=(5, 5))
    cm_train = ConfusionMatrixDisplay.from_estimator(
        model, X_train, y_train, values_format="d", ax = ax
    )
    cm_train.ax_.set_title("Training confusion matrix for " +
model_name)
    fig, ax = plt.subplots(figsize=(5, 5))
    cm_test = ConfusionMatrixDisplay.from_estimator(
        model, X_test, y_test, values_format="d", ax = ax
    )
    cm_test.ax_.set_title("Test confusion matrix for " + model_name)
```

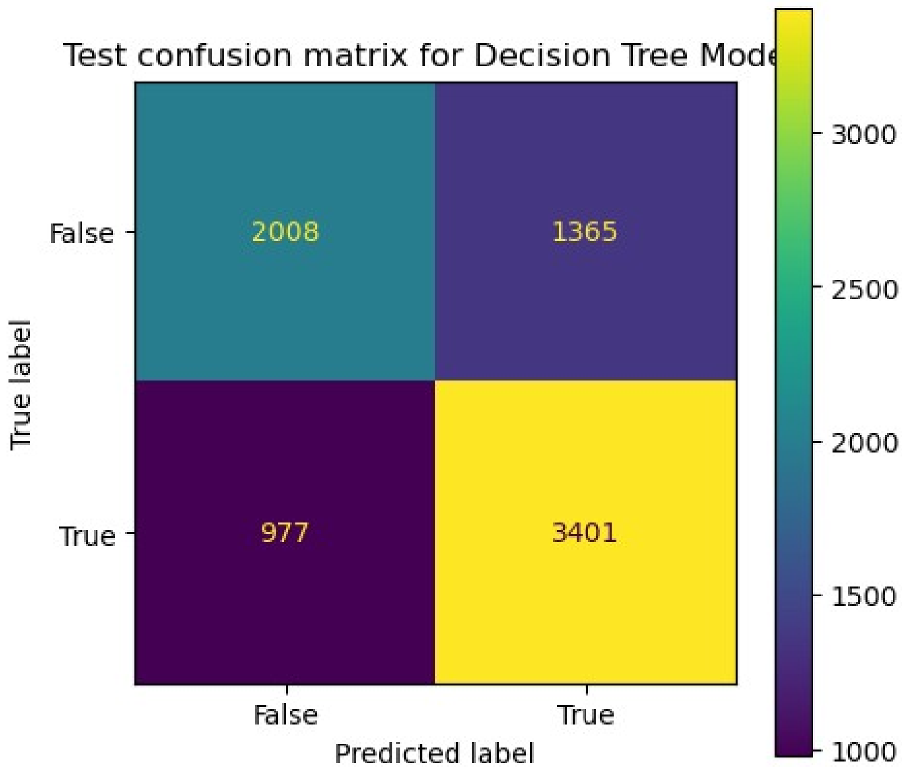


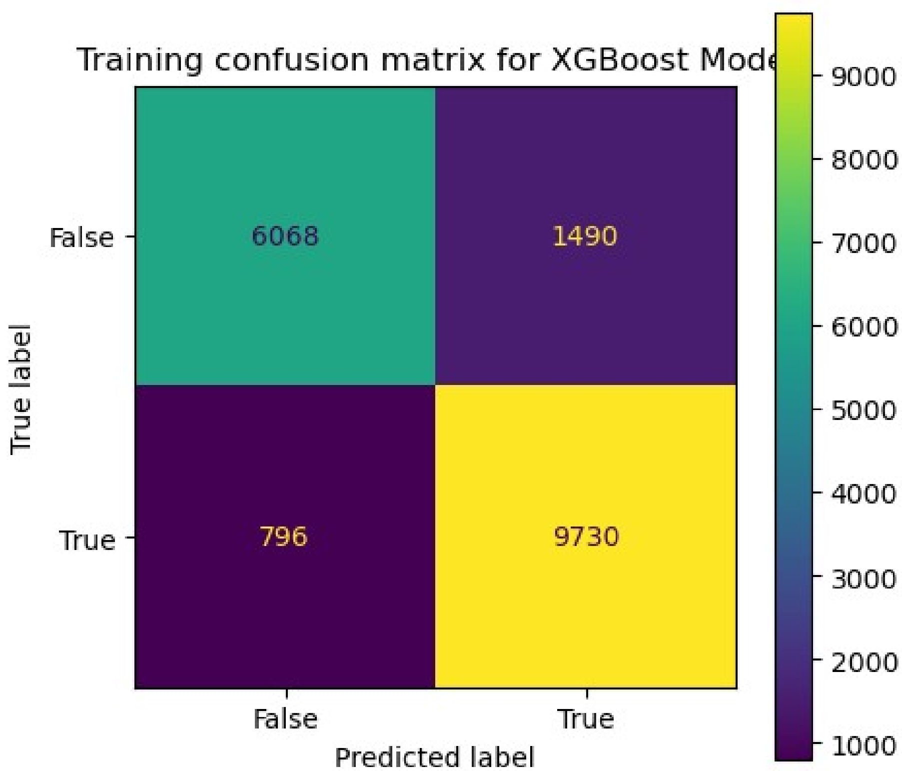

Test confusion matrix for LogReg Model
True label
Predicted label
False
True
False
True
2334
1039
1271
3107
3000
2750
2500
2250
2000
1750
1500
1250

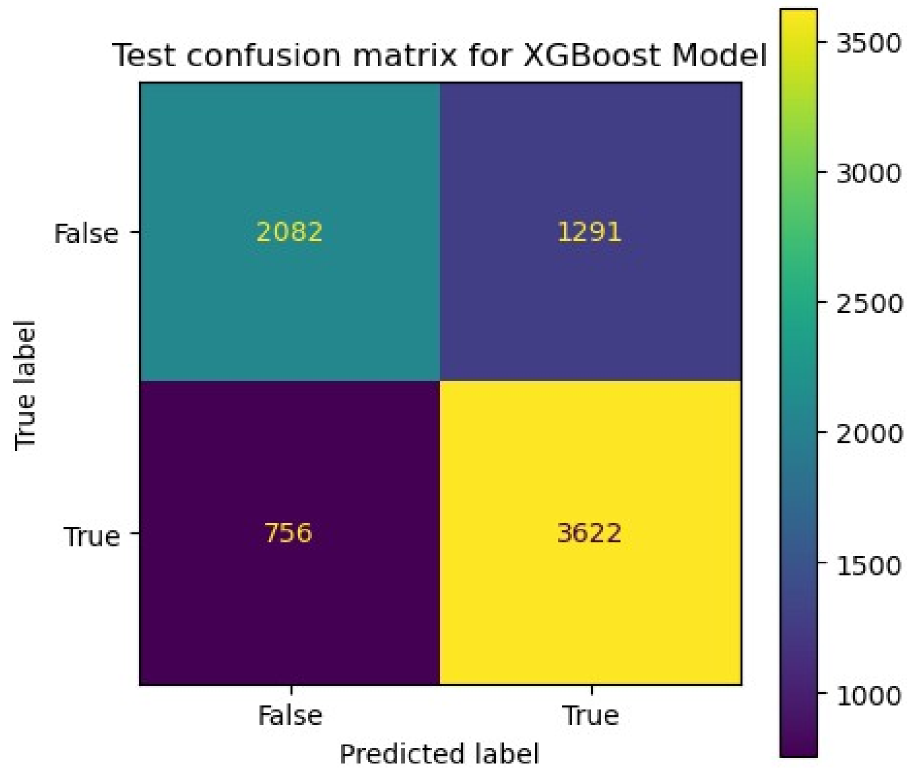

Training confusion matrix for Random Forest Model
True label
Predicted label
False
True
False
True
7551
7
54
10472
10000
8000
6000
4000
2000


Test confusion matrix for Random Forest Mod
True label
Predicted label
False
True
False
True
1986
1387
789
3589
3500
3000
2500
2000
1500
1000


Training confusion matrix for Decision Tree Mo
True label
Predicted label
False
True
False
True
4920
2638
2089
8437
8000
7000
6000
5000
4000
3000


Test confusion matrix for Decision Tree Mode
True label
False
True
2008
1365
977
3401
False
True
Predicted label
3000
2500
2000
1500
1000


Training confusion matrix for XGBoost Mode
True label
False
True
6068
1490
796
9730
False
True
Predicted label
9000
8000
7000
6000
5000
4000
3000
2000
1000



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?
  - Given the problem, minimizing the risk of both false positives and negatives would be the most ideal, as false positives would cause individuals who wish to atone to be treated unjustly, and false negatives would lead to unrepentant criminals being released and allowed to recidivate. While each model provides relatively similar testing results, `logreg_model` and `tree_model` both provide smaller differences in training and testing scores, indicating that they are less overfitted to the dataset. In contrast, `both xgboost_model` and `rf_model` show a very big gap between the training accuracy and testing accuracy, which is an indicator of overfitting. `logreg_model` provides a better testing `roc_auc_score` and less false positives because it has better precision. In contrast `tree_model` provides better training and testing `f1 score` and fewer false negatives because it has a higher recall. Given that our main focus is on the positive class of `Recidivism_Within_3years`, a greater `f1 score` and `recall` would be more important, and thus we would consider the classifier `tree_model` to be the most effective deployment model.
- Which classifier is the most "severe" (a.k.a. classifies more people as at risk of committing another crime within 3 years)?
  - The most "severe" classifier is the one that provides the greatest proportion of True predictions, regardless of whether they are actually True or False. According

    to the testing confusion matrices, the Random Forest Model (`rf_model`) provides the greatest proportion of True predictions and thus is the most "severe".
- Which classifier is the most cautious (a.k.a. classifies fewer people as at risk of committing another crime within 3 years)?
    - The most cautious classifier is the one that provides the greatest proportion of False predictions, regardless of whether they are actually True or False. According to the testing confusion matrices, the LogReg Model (`logreg_model`) provides the greatest proportion of False predictions and thus is the most cautious.
- Czakon, J. (2024, September 10). F1 score vs ROC AUC vs Accuracy Vs PR AUC: Which evaluation metric should you choose?. neptune.ai. https://neptune.ai/blog/f1-score-accuracy-roc-auc-pr-auc

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.

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	$\frac{TP_b+FP_b}{TP_w+FP_w}$
PPGRD	$\frac{(TP_b+FP_b)/(TP_b+FP_b+TN_b+FN_b)}{(TP_w+FP_w)/(TP_w+FP_w+TN_w+FN_w)}=$
FDRD	$\frac{(FP_b)/(TP_b+FP_b)}{(FP_w)/(TP_w+FP_w)}=\frac{(FP_b)(TP_w+FP_w)}{(FP_w)(TP_b+FP_b)}$
FPRD	$\frac{(FP_b)/(TN_b+FP_b)}{(FP_w)/(TN_w+FP_w)}=\frac{(FP_b)(TN_w+FP_w)}{(FP_w)(TN_b+FP_b)}$
FORD	$\frac{(FN_b)/(TN_b+FN_b)}{(FN_w)/(TN_w+FN_w)}=\frac{(FN_b)(TN_w+FN_w)}{(FN_w)(TN_b+FN_b)}$
FNRD	$\frac{(FN_b)/(TP_b+FN_b)}{(FN_w)/(TP_w+FN_w)}=\frac{(FN_b)(TP_w+FN_w)}{(FN_w)(TP_b+FN_b)}$

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.
train_df_b = train_df[train_df["Race"] == "BLACK"]
train_df_w = train_df[train_df["Race"] == "WHITE"]
test_df_b = test_df[test_df["Race"] == "BLACK"]
test_df_w = test_df[test_df["Race"] == "WHITE"]
# train_df_b.head()

# Creating training and test sets and separating features and target
```

```
X_train_b, y_train_b = (
    train_df_b.drop(columns=["Recidivism_Within_3years"]),
    train_df_b["Recidivism_Within_3years"],
)
X_test_b, y_test_b = (
    test_df_b.drop(columns=["Recidivism_Within_3years"]),
    test_df_b["Recidivism_Within_3years"],
)
X_train_w, y_train_w = (
    train_df_w.drop(columns=["Recidivism_Within_3years"]),
    train_df_w["Recidivism_Within_3years"],
)
X_test_w, y_test_w = (
    test_df_w.drop(columns=["Recidivism_Within_3years"]),
    test_df_w["Recidivism_Within_3years"],
)

# https://datascience.stackexchange.com/questions/28493/confusion-
matrix-get-items-fp-fn-tp-tn-python

fairness_mets = {
    "model": [],
    # "PPRD": [],
    # "PPGRD": [],
    # "FDRD": [],
    # "FPRD": [],
    # "FORD": [],
    # "FNRD": [],
    "PPRD_adfr": [],
    "PPGRD_adfr": [],
    "FDRD_adfr": [],
    "FPRD_adfr": [],
    "FORD_adfr": [],
    "FNRD_adfr": [],
}

def fairness_metrics(modelname, model, X_b, y_b, X_w, y_w, name):
    # cm = ConfusionMatrixDisplay.from_estimator(
    #     model, X_train, y_train, values_format="d"
    # ).confusion_matrix

    cm_b = confusion_matrix(y_b, model.predict(X_b))
    cm_w = confusion_matrix(y_w, model.predict(X_w))

    TP_b = cm_b[0][0]
    FP_b = cm_b[0][1]
    FN_b = cm_b[1][0]
    TN_b = cm_b[1][1]

    TP_w = cm_w[0][0]
```

```
    FP_w = cm_w[0][1]
    FN_w = cm_w[1][0]
    TN_w = cm_w[1][1]

    PPRD = (TP_b + FP_b)/(TP_w + FP_w)
    PPGRD = ((TP_b + FP_b)/(TP_b + FP_b + FN_b + TN_b))/((TP_w +
FP_w)/(TP_w + FP_w + FN_w + TN_w))
    FDRD = (FP_b/(TP_b + FP_b))/(FP_w/(TP_w + FP_w))
    FPRD = (FP_b/(TN_b + FP_b))/(FP_w/(TN_w + FP_w))
    FORD = (FN_b/(TN_b + FN_b))/(FN_w/(TN_w + FN_w))
    FNRD = (FN_b/(TP_b + FN_b))/(FN_w/(TP_w + FN_w))

    PPRD_adfr = abs(PPRD - 1)
    PPGRD_adfr = abs(PPGRD - 1)
    FDRD_adfr = abs(FDRD - 1)
    FPRD_adfr = abs(FPRD - 1)
    FORD_adfr = abs(FORD - 1)
    FNRD_adfr = abs(FNRD - 1)

    fairness_mets["model"].append(str(modelname) + " (" + str(name) +
")")
    # fairness_mets["PPRD"].append(PPRD)
    # fairness_mets["PPGRD"].append(PPGRD)
    # fairness_mets["FDRD"].append(FDRD)
    # fairness_mets["FPRD"].append(FPRD)
    # fairness_mets["FORD"].append(FORD)
    # fairness_mets["FNRD"].append(FNRD)
    fairness_mets["PPRD_adfr"].append(PPRD_adfr)
    fairness_mets["PPGRD_adfr"].append(PPGRD_adfr)
    fairness_mets["FDRD_adfr"].append(FDRD_adfr)
    fairness_mets["FPRD_adfr"].append(FPRD_adfr)
    fairness_mets["FORD_adfr"].append(FORD_adfr)
    fairness_mets["FNRD_adfr"].append(FNRD_adfr)

# modeldict = {"logreg_model": logreg_model, "rf_model": rf_model,
"tree_model": tree_model, "xgboost_model": xgboost_model}
# modellist = ["logreg_model", "rf_model", "tree_model",
"xgboost_model"]

# for modelname in modellist:
#     fairness_metrics(modelname, modeldict[modelname], X_train_b,
y_train_b, X_train_w, y_train_w, "Training")
#     fairness_metrics(modelname, modeldict[modelname], X_test_b,
y_test_b, X_test_w, y_test_w, "Testing")

models = [logreg_model, rf_model, tree_model, xgboost_model]
model_names = ["LogReg Model", "Random Forest Model", "Decision Tree
Model ", "XGBoost Model"]
for model, model_name in zip(models,model_names):
    fairness_metrics(model_name, model, X_train_b, y_train_b,
```

```
X_train_w, y_train_w, "Training")
    fairness_metrics(model_name, model, X_test_b, y_test_b, X_test_w,
y_test_w, "Testing")

fairness_mets_df = pd.DataFrame(fairness_mets)
fairness_mets_df = fairness_mets_df.set_index(["model"])
fairness_mets_df

                                 PPRD_adfr  PPGRD_adfr  FDRD_adfr
FPRD_adfr  \
model

LogReg Model (Training)           0.271716    0.052716   0.213997
0.060968
LogReg Model (Testing)            0.294558    0.056298   0.166160
0.027451
Random Forest Model (Training)    0.271716    0.052716   0.965847
0.792561
Random Forest Model (Testing)     0.294558    0.056298   0.179099
0.022388
Decision Tree Model  (Training)   0.271716    0.052716   0.225651
0.060007
Decision Tree Model  (Testing)    0.294558    0.056298   0.213249
0.043059
XGBoost Model (Training)          0.271716    0.052716   0.320936
0.156987
XGBoost Model (Testing)           0.294558    0.056298   0.109142
0.000879

                                 FORD_adfr  FNRD_adfr
model
LogReg Model (Training)           0.059559   0.072639
LogReg Model (Testing)            0.035063   0.091346
Random Forest Model (Training)    0.319480   0.445670
Random Forest Model (Testing)     0.130365   0.056076
Decision Tree Model  (Training)   0.126143   0.046944
Decision Tree Model  (Testing)    0.102355   0.086773
XGBoost Model (Training)          0.187694   0.041546
XGBoost Model (Testing)           0.002636   0.132486
```

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?

- `xgboost_model` exhibits the least amount of bias, providing the least amount of distance from the reference for `FDRD_adfr`, `FPRD_adfr`, and `FORD_adfr`, though it provides the most amount of distance from the reference for `FNRD_adfr`.
- `tree_model` exhibits the most amount of bias, providing the most amount of distance from the reference for `FDRD_adfr`, `FPRD_adfr`, and `FNRD_adfr`, while providing the second greatest amount of distance from the reference for `FORD_adfr`.
- Given that the application is for the the identification of across populations of race, measurement bias will likely occur to the detriment of black defendants. As such, the fairness metrics `FDRD_adfr` and `FPRD_adfr` should be minimised to reduce both the likelihood of black defendants being predicted as true in general and the likelihood of black defendants being predicted as guilty when they are actually innocent. As a consequence, the fairness metrics `FORD_adfr` and `FNRD_adfr` could be taken to less consideration.
- Given the fairness results, in terms of minimising bias, the model `xgboost_model` would be the most optimal for this application.

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.

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.

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_nam
es_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                         0.777355
pipeline-2__Age_at_Release_18-22                         0.769491
pipeline-2__Delinquency_Reports_1                        0.635838
pipeline-2__Age_at_Release_23-27                         0.488772
pipeline-2__Prior_Arrest_Episodes_Felony_0               0.473405
pipeline-2__Gender_M                                     0.458260
passthrough__Prior_Revocations_Parole                    0.362398
passthrough__Condition_MH_SA                             0.359117
pipeline-1__Jobs_Per_Year                                0.312944
pipeline-2__Prison_Years_Less than 1 year                0.307459
```

```
coeff_df_sorted.tail(10)
```

```
                                                              Coefficient
pipeline-2__Delinquency_Reports_3                               -0.205489
pipeline-2__Prior_Arrest_Episodes_PPViolationCh...              -0.216048
pipeline-2__Age_at_Release_38-42                                -0.235466
pipeline-2__Prior_Arrest_Episodes_Felony_2                      -0.249548
pipeline-2__Age_at_Release_43-47                                -0.350821
pipeline-2__Program_Attendances_10 or more                      -0.385809
pipeline-2__Prior_Arrest_Episodes_Felony_1                      -0.501982
pipeline-2__Delinquency_Reports_4 or more                       -0.507616
pipeline-1__Percent_Days_Employed                               -0.663686
pipeline-2__Age_at_Release_48 or older                          -0.752269
```

The features `pipeline-2__Gang_Affiliated_True`, `pipeline-2__Age_at_Release_18-22`, `pipeline-2__Delinquency_Reports_1`, `pipeline-2__Age_at_Release_23-27`, `pipeline-2__Prior_Arrest_Episodes_Felony_0`, `pipeline-2__Gender_M`, `passthrough__Prior_Revocations_Parole`, `passthrough__Condition_MH_SA`, `pipeline-1__Jobs_Per_Year` and `pipeline-2__Prison_Years_Less than 1 year` bring the prediction towards the positive class. The features `pipeline-2__Delinquency_Reports_3`, `pipeline-2__Prior_Arrest_Episodes_PPViolationCharges_0`, `pipeline-2__Age_at_Release_38-42`, `pipeline-2__Prior_Arrest_Episodes_Felony_2`, `pipeline-2__Age_at_Release_43-47`, `pipeline-2__Program_Attendances_10 or more`, `pipeline-2__Prior_Arrest_Episodes_Felony_1`, `pipeline-2__Delinquency_Reports_4 or more`, `pipeline-1__Percent_Days_Employed`, and `pipeline-2__Age_at_Release_48 or older` bring the prediction towards the negative class. The coefficients that may be unfairly influencing the decision are `pipeline-2__Age_at_Release_18-22`, `pipeline-2__Age_at_Release_23-27`, `pipeline-2__Prior_Arrest_Episodes_Felony_0`, `pipeline-2__Gender_M`, `passthrough__Prior_Revocations_Parole`, `passthrough__Condition_MH_SA`, `pipeline-2__Prior_Arrest_Episodes_PPViolationCharges_0`, `pipeline-2__Age_at_Release_38-42`, `pipeline-2__Prior_Arrest_Episodes_Felony_2`, `pipeline-2__Age_at_Release_43-47`, `pipeline-2__Program_Attendances_10 or more`, `pipeline-2__Prior_Arrest_Episodes_Felony_1`, and `pipeline-2__Age_at_Release_48 or older`.

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]
```

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:

```
# Your answer here
display(y_test[106:107])
display(logreg_model.predict(hard_sample)) # Prediction of logistic regression model: False
display(hard_sample.T)
```

```
106    False
Name: Recidivism_Within_3years, dtype: bool
```

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

```
106
Unnamed: 0
5645
ID
5788
Gender
F
Race
WHITE
Age_at_Release                                              48 or
older
Residence_PUMA
3
Gang_Affiliated
NaN
Supervision_Risk_Score_First
6.0
Supervision_Level_First
High
Education_Level                                     At least some
college
Dependents
0
Prison_Offense
Drug
Prison_Years                                    Greater than 2 to 3
```

```
years
Prior_Arrest_Episodes_Felony
7
Prior_Arrest_Episodes_Misd
3
Prior_Arrest_Episodes_Violent
0
Prior_Arrest_Episodes_Property                                             5
or more
Prior_Arrest_Episodes_Drug
3
Prior_Arrest_Episodes_PPViolationCharges
1
Prior_Arrest_Episodes_DVCharges
False
Prior_Arrest_Episodes_GunCharges
False
Prior_Conviction_Episodes_Felony
1
Prior_Conviction_Episodes_Misd
2
Prior_Conviction_Episodes_Viol
False
Prior_Conviction_Episodes_Prop                                             3
or more
Prior_Conviction_Episodes_Drug
1
Prior_Conviction_Episodes_PPViolationCharges
True
Prior_Conviction_Episodes_DomesticViolenceCharges
False
Prior_Conviction_Episodes_GunCharges
False
Prior_Revocations_Parole
False
Prior_Revocations_Probation
False
Condition_MH_SA
True
Condition_Cog_Ed
False
Condition_Other
False
Violations_ElectronicMonitoring
False
Violations_Instruction
False
Violations_FailToReport
False
```

```
Violations_MoveWithoutPermission
True
Delinquency_Reports
0
Program_Attendances
7
Program_UnexcusedAbsences
0
Residence_Changes
1
Avg_Days_per_DrugTest
NaN
DrugTests_THC_Positive
NaN
DrugTests_Cocaine_Positive
NaN
DrugTests_Meth_Positive
NaN
DrugTests_Other_Positive
NaN
Percent_Days_Employed
0.596215
Jobs_Per_Year
2.0
Employment_Exempt
False
Recidivism_Arrest_Year1
False
Recidivism_Arrest_Year2
False
Recidivism_Arrest_Year3
False
Training_Sample
0
```

`hard_sample` is predicted with `Recidivism_Within_3years == 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?

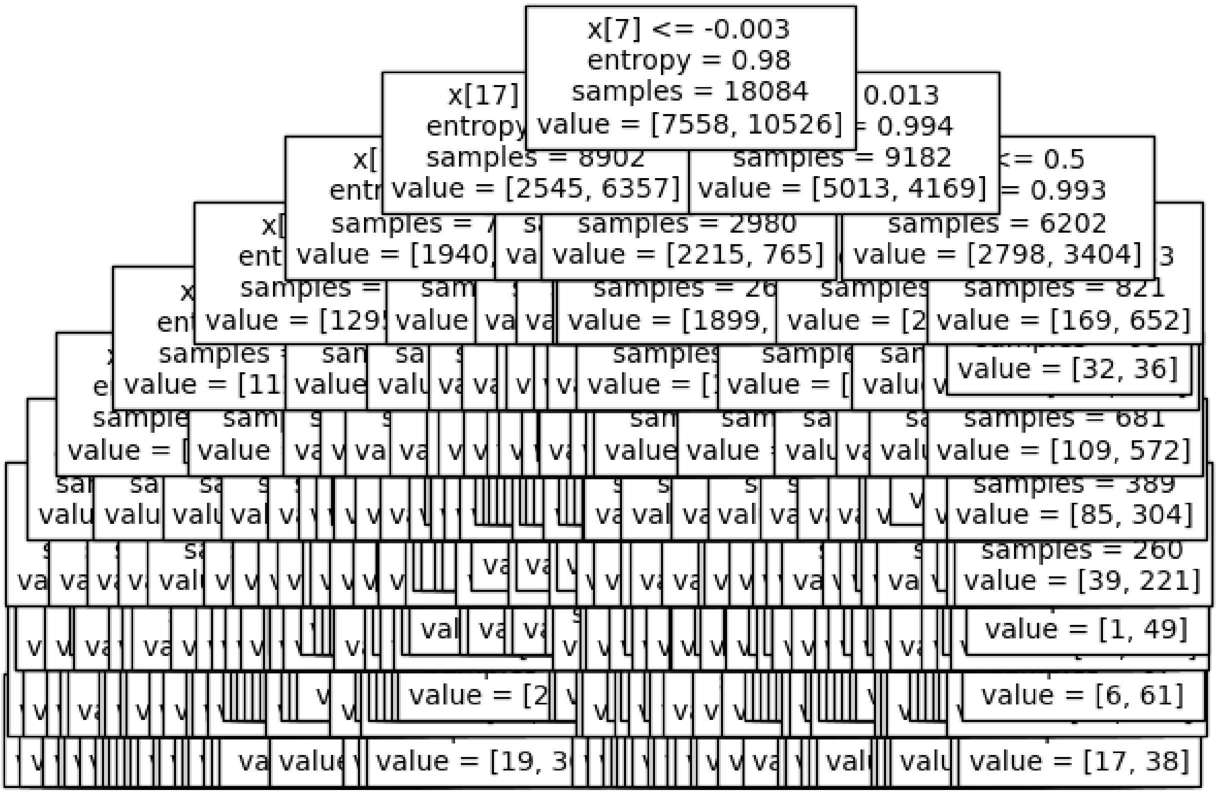
The feature `pipeline-2__Age_at_Release_48 or older` appears to have contributed the most to the final prediction of `Recidivism_Within_3years == False`. The feature `passthrough__Condition_MH_SA` pushed the most in the opposite direction.

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>
```

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__Jobs_Per_Year <= -
```

```
0.37
|   |   |   |   |   |   |   |   |   |--- class: True
|   |   |   |   |   |   |   |   |--- pipeline-1__Jobs_Per_Year >  -
0.37
|   |   |   |   |   |   |   |   |   |--- 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
```

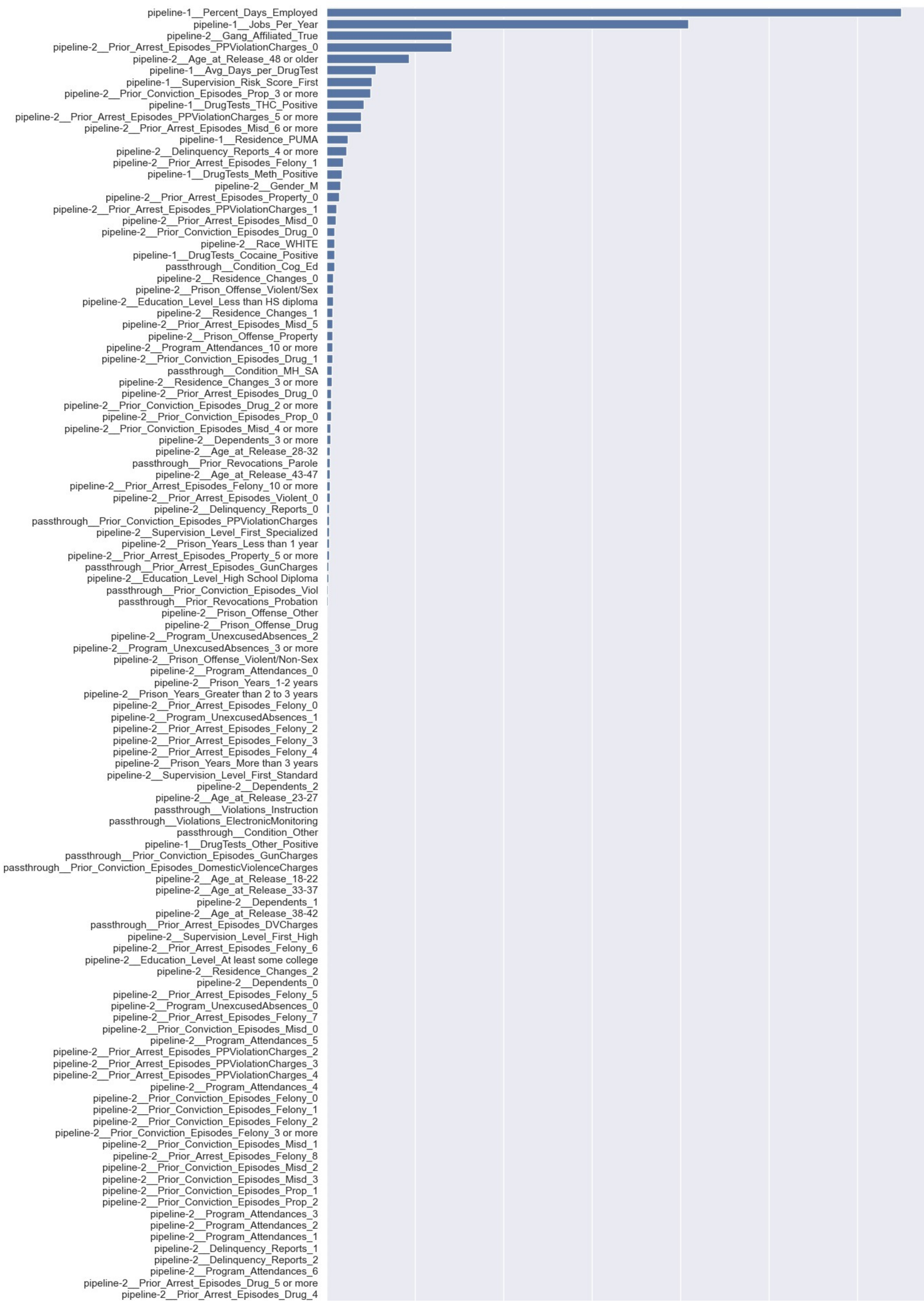
```
|   |   |   |   |   |   |   |   |--- class: False
|   |   |   |   |   |   |--- pipeline-2__Gender_M >  0.50
|   |   |   |   |   |   |   |--- pipeline-
1__Supervision_Risk_Score_First <= -0.67
|   |   |   |   |   |   |   |   |--- pipeline-2__Race_WHITE <= 0.50
|   |   |   |   |   |   |   |   |   |--- 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
```

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

According to the features importance of `tree_model`, the features `pipeline-1__Percent_Days_Employed`, `pipeline-1__Jobs_Per_Year`, `pipeline-2__Gange_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` and `pipeline-2__Prior_Arrest_Episodes_Misd_6 or more` contribute the most to the classification of samples in the model. Of these features, `pipeline-1__Percent_Days_Employed`, `pipeline-1__Jobs_Per_Year`, `pipeline-2__Gange_Affiliated_True`, `pipeline-2__Prior_Arrest_Episodes_PPViolationCharges_0`, and `pipeline-2__Age_at_Release_48 or older` are among the features in the logistic regression model with the most extreme coefficients as shown in Q7. `pipeline-1__Percent_Days_Employed` and `pipeline-2__Gange_Affiliated_True` provide very strong contributions to the prediction for both models, `pipeline-1__Jobs_Per_Year` and `pipeline-2__Prior_Arrest_Episodes_PPViolationCharges_0` provide relatively stronger contributions to `tree_model`, while `pipeline-2__Age_at_Release_48 or older` provides a relatively stronger contribution to `logreg_model`. A big limitation of feature importance compared to observing coefficients is that it does not provide a direct measure of how the features will influence the model prediction (e.g. Push prediction to positive or negative class), making it more difficult to interpret in comparison.

- https://www.codecademy.com/article/fe-feature-importance-final

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: The classification is incorrect
display(y_test[106:107]) # False
display(tree_model.predict(hard_sample)) # True
```

```
106    False
Name: Recidivism_Within_3years, dtype: bool

array([ True])
```

The decision tree model predicts it as `True`, which means the prediction on the `hard_sample` 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.

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

# Step 2: train model on random forest predictions on the training set
surrogate_model_rf.fit(X_train, tree_model.predict(X_train))

# Step 3: visualize weights of surrogate model, as we did for the original logistic regression model
s_feature_names = np.array(surrogate_model_rf.named_steps['columntransformer'].get_feature_names_out())
s_coeffs = surrogate_model_rf.named_steps["logisticregression"].coef_.flatten()
s_coeff_df = pd.DataFrame(s_coeffs, index=s_feature_names, columns=["Coefficient"])
s_coeff_df_sorted = s_coeff_df.sort_values(by="Coefficient", ascending=False)
display(s_coeff_df_sorted.head(10))
display(s_coeff_df_sorted.tail(10))
```

	Coefficient
pipeline-2__Gang_Affiliated_True	1.981884
pipeline-2__Prior_Arrest_Episodes_PPViolationCh...	1.090830
pipeline-1__Jobs_Per_Year	0.626207
pipeline-2__Age_at_Release_18-22	0.592761
pipeline-2__Age_at_Release_23-27	0.510936
pipeline-2__Gender_M	0.441204
pipeline-2__Delinquency_Reports_1	0.431845
pipeline-2__Prior_Conviction_Episodes_Prop_3 or...	0.425607

```
pipeline-2__Prior_Arrest_Episodes_Misd_6 or more       0.346842
pipeline-2__Residence_Changes_3 or more                0.346202

                                                    Coefficient
pipeline-2__Prior_Conviction_Episodes_Prop_1          -0.180564
pipeline-2__Program_Attendances_10 or more            -0.183690
pipeline-2__Prison_Years_More than 3 years            -0.196347
pipeline-2__Prior_Arrest_Episodes_Felony_2            -0.212103
pipeline-2__Prior_Arrest_Episodes_Misd_0              -0.274397
pipeline-2__Delinquency_Reports_4 or more             -0.575022
pipeline-2__Prior_Arrest_Episodes_Felony_1            -0.634692
pipeline-2__Prior_Arrest_Episodes_PPViolationCh...    -0.949615
pipeline-2__Age_at_Release_48 or older                -1.330968
pipeline-1__Percent_Days_Employed                     -1.743467

display(

set(coeff_df_sorted.head(10).index.values.tolist()).intersection(s_coe
ff_df_sorted.head(10).index.values.tolist())
    # - (set(coeff_df_sorted.head(10).index.values.tolist()) -
set(s_coeff_df_sorted.head(10).index.values.tolist()))
)
display(

set(coeff_df_sorted.tail(10).index.values.tolist()).intersection(s_coe
ff_df_sorted.tail(10).index.values.tolist())
    # - (set(coeff_df_sorted.tail(10).index.values.tolist()) -
set(s_coeff_df_sorted.tail(10).index.values.tolist()))
)

{'pipeline-1__Jobs_Per_Year',
 'pipeline-2__Age_at_Release_18-22',
 'pipeline-2__Age_at_Release_23-27',
 'pipeline-2__Delinquency_Reports_1',
 'pipeline-2__Gang_Affiliated_True',
 'pipeline-2__Gender_M'}

{'pipeline-1__Percent_Days_Employed',
 'pipeline-2__Age_at_Release_48 or older',
 'pipeline-2__Delinquency_Reports_4 or more',
 'pipeline-2__Prior_Arrest_Episodes_Felony_1',
 'pipeline-2__Prior_Arrest_Episodes_Felony_2',
 'pipeline-2__Prior_Arrest_Episodes_PPViolationCharges_0',
 'pipeline-2__Program_Attendances_10 or more'}
```

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?

The features stored in `s_coeff_df_sorted.head(10)` are the top 10 features in the random forest model that push the prediction toward the positive class while the features stored in `s_coeff_df_sorted.tail(10)` are the top 10 features in the random forest model that push the prediction toward the negative class. The features stored in `logreg_rf_positive_coef_intersection` above are the ones that appear to be the top 10 features that push the prediction toward the positive class in both `logreg_model` and `surrogate_model_rf`, while the features stored in `logreg_rf_negative_coef_intersection` above are the ones that appear to be the top 10 features that push the prediction toward the negative class in both `logreg_model` and `surrogate_model_rf`. In `surrogate_model_rf`, the features `pipeline-1__Percent_Days_Employed`, `pipeline-2__Age_at_Release_48 or older`, `pipeline-2__Prior_Arrest_Episodes_PPViolationCharges_0`, `pipeline-2__Gange_Affiliated_True`, `pipeline-1__Jobs_Per_Year` and `pipeline-2__Prior_Arrest_Episodes_PPViolationCharges_5 or more` are the primary characteristics that influence the prediction, as `surrogate_model_rf` has significant increases in their corresponding **absolute weight values** compared to the **absolute weights** of `logreg_model`. `pipeline-1__Percent_Days_Employed`, `pipeline-2__Age_at_Release_48 or older` and `pipeline-2__Prior_Arrest_Episodes_PPViolationCharges_0` push the prediction towards `Recidivism_Within_3years == False`, while `pipeline-2__Gange_Affiliated_True`, `pipeline-1__Jobs_Per_Year` and `pipeline-2__Prior_Arrest_Episodes_PPViolationCharges_5 or more` push the prediction towards `Recidivism_Within_3years = True`. The greater weights of `surrogate_model_rf` correspond to the higher levels of feature importances indicated by the bar plot of Gini importance.

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
def get_num_correct(y, y_pred, t=0.5):
    y_correct = np.array([0.0 if p < t else 1.0 for p in y_pred])
    return sum([1.0 for p, p_pred in zip(y, y_correct) if p ==
p_pred])
```

```
def count_rsquare(y, y_pred, t=0.5):
    n = float(len(y))
    num_correct = get_num_correct(y, y_pred, t)
    return num_correct / n

# 0.7043795620437956 of the variance is captured by the surrogate
model
# count_rsquare(y_train, surrogate_model_rf.predict(X_train))
display(count_rsquare(rf_model.predict(X_train),
surrogate_model_rf.predict(X_train)), # 0.7077527095775271
        count_rsquare(rf_model.predict(X_test),
surrogate_model_rf.predict(X_test))) # 0.8472455167075216
```

```
0.7077527095775271
```

```
0.8472455167075216
```

The R2 value between the Random Forest Model predictions and the surrogate model predictions on the training set is 0.7077527095775271; the surrogate model explains 70.775% (5 s.f.) of the variance in the Random Forest model's predictions on the training set. The R2 value between the Random Forest Model predictions and the surrogate model predictions on the testing set is 0.8472455167075216; the surrogate model explains 84.725% (5 s.f.) of the variance in the Random Forest model's predictions on the testing set.

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
# 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_xgboost =
joblib.load("models_for_A3/NIJ_logreg.joblib")

# Step 2: train model on random forest predictions on the training set
surrogate_model_xgboost.fit(X_train, xgboost_model.predict(X_train))
```

```
# Step 3: visualize weights of surrogate model, as we did for the
original logistic regression model
s1_feature_names =
np.array(surrogate_model_xgboost.named_steps['columntransformer'].get_
feature_names_out())
s1_coeffs =
surrogate_model_xgboost.named_steps["logisticregression"].coef_.flatte
n()
s1_coeff_df = pd.DataFrame(s1_coeffs, index=s1_feature_names,
columns=["Coefficient"])
s1_coeff_df_sorted = s1_coeff_df.sort_values(by="Coefficient",
ascending=False)
display(s1_coeff_df_sorted.head(10))
display(s1_coeff_df_sorted.tail(10))
```

	Coefficient
pipeline-2__Age_at_Release_18-22	1.344978
pipeline-2__Gang_Affiliated_True	1.322722
pipeline-2__Gender_M	0.939579
pipeline-2__Delinquency_Reports_1	0.853040
pipeline-2__Age_at_Release_23-27	0.767391
pipeline-2__Prior_Arrest_Episodes_Felony_0	0.668436
passthrough__Condition_MH_SA	0.643236
pipeline-2__Prior_Arrest_Episodes_Felony_10 or ...	0.606695
pipeline-2__Prison_Years_Less than 1 year	0.591937
passthrough__Violations_ElectronicMonitoring	0.589146

	Coefficient
pipeline-2__Prison_Years_Greater than 2 to 3 years	-0.373178
pipeline-2__Prior_Arrest_Episodes_Felony_2	-0.378917
pipeline-2__Age_at_Release_38-42	-0.383271
pipeline-2__Prior_Arrest_Episodes_PPViolationCh...	-0.505892
pipeline-2__Age_at_Release_43-47	-0.568909
pipeline-2__Program_Attendances_10 or more	-0.747395
pipeline-2__Delinquency_Reports_4 or more	-0.860045
pipeline-2__Prior_Arrest_Episodes_Felony_1	-0.914729
pipeline-2__Age_at_Release_48 or older	-1.321597
pipeline-1__Percent_Days_Employed	-1.326048

```
display(

set(coeff_df_sorted.head(10).index.values.tolist()).intersection(s1_co
eff_df_sorted.head(10).index.values.tolist())
    # - (set(coeff_df_sorted.head(10).index.values.tolist()) -
set(s1_coeff_df_sorted.head(10).index.values.tolist()))
)
display(

set(coeff_df_sorted.tail(10).index.values.tolist()).intersection(s1_co
eff_df_sorted.tail(10).index.values.tolist())
```

```
    # - (set(coeff_df_sorted.tail(10).index.values.tolist()) -
set(s1_coeff_df_sorted.tail(10).index.values.tolist()))
)
```

```
{'passthrough__Condition_MH_SA',
 'pipeline-2__Age_at_Release_18-22',
 'pipeline-2__Age_at_Release_23-27',
 'pipeline-2__Delinquency_Reports_1',
 'pipeline-2__Gang_Affiliated_True',
 'pipeline-2__Gender_M',
 'pipeline-2__Prior_Arrest_Episodes_Felony_0',
 'pipeline-2__Prison_Years_Less than 1 year'}
```

```
{'pipeline-1__Percent_Days_Employed',
 'pipeline-2__Age_at_Release_38-42',
 'pipeline-2__Age_at_Release_43-47',
 'pipeline-2__Age_at_Release_48 or older',
 'pipeline-2__Delinquency_Reports_4 or more',
 'pipeline-2__Prior_Arrest_Episodes_Felony_1',
 'pipeline-2__Prior_Arrest_Episodes_Felony_2',
 'pipeline-2__Prior_Arrest_Episodes_PPViolationCharges_0',
 'pipeline-2__Program_Attendances_10 or more'}
```

The features stored in `s1_coeff_df_sorted.head(10)` are the top 10 features in the XGBoost model that push the prediction toward the positive class while the features stored in `s1_coeff_df_sorted.tail(10)` are the top 10 features in the XGBoost model that push the prediction toward the negative class. The features stored in `logreg_xgboost_positive_coef_intersection` above are the ones that appear to be the top 10 features that push the prediction toward the positive class in both `logreg_model` and `surrogate_model_xgboost`, while the features stored in `logreg_xgboost_negative_coef_intersection` above are the ones that appear to be the top 10 features that push the prediction toward the negative class in both `logreg_model` and `surrogate_model_xgboost`. In `surrogate_model_xgboost`, the features `pipeline-1__Percent_Days_Employed`, `pipeline-2__Age_at_Release_48 or older`, `pipeline-2__Prior_Arrest_Episodes_Felony_1`, `pipeline-2__Delinquency_Reports_4 or more`, `pipeline-2__Program_Attendances_10 or more`, `pipeline-2__Age_at_Release_18-22`, `pipeline-2__Gange_Affiliated_True`, `pipeline-2__Gender_M`, `pipeline-2__Delinquency_Reports_1`, `pipeline-2__Age_at_Release_23-27`, are the primary characteristics that influence the prediction, as `surrogate_model_xgboost` has significant increases in their corresponding absolute weight values compared to the weights of `logreg_model`. `pipeline-1__Percent_Days_Employed`, `pipeline-2__Age_at_Release_48 or older`, `pipeline-2__Prior_Arrest_Episodes_Felony_1`, `pipeline-2__Delinquency_Reports_4 or more`, and `pipeline-2__Program_Attendances_10 or more` push the prediction towards `Recidivism_Within_3years == False`, while `pipeline-2__Age_at_Release_18-22`, `pipeline-2__Gange_Affiliated_True`, `pipeline-2__Gender_M`, `pipeline-2__Delinquency_Reports_1` and `pipeline-2__Age_at_Release_23-27` push the prediction towards `Recidivism_Within_3years =`

`True`. The greater weights of `surrogate_model_xgboost` correspond to the higher levels of feature importances indicated by the bar plot of Gini importance. In general, the features `pipeline-1__Percent_Days_Employed`, `pipeline-2__Age_at_Release_48 or older`, `pipeline-2__Age_at_Release_18-22` and `pipeline-2__Gange_Affiliated_True` and their respective effects on model prediction are common as particularly important features between `logreg_model`, `tree_model` and `xgboost_model`.

```
# # 0.714609599646096 of the variance is captured by the log_reg model
# display(count_rsquare(y_train, logreg_model.predict(X_train)))

# # 0.7167109046671091 of the variance is captured by the surrogate
model
# display(count_rsquare(y_train,
surrogate_model_xgboost.predict(X_train)))

display(count_rsquare(xgboost_model.predict(X_train),
surrogate_model_xgboost.predict(X_train)), # 0.8152510506525105
        count_rsquare(xgboost_model.predict(X_test),
surrogate_model_xgboost.predict(X_test))) # 0.8440201264352987
```

```
0.8152510506525105

0.8440201264352987
```

The R2 value between the XGBoost Model predictions and the surrogate model predictions on the training set is 0.8152510506525105; the surrogate model explains 81.525% (5 s.f.) of the variance in the XGBoost model's predictions on the training set. The R2 value between the XGBoost Model predictions and the surrogate model predictions on the testing set is 0.8440201264352987; the surrogate model explains 84.440% (5 s.f.) of the variance in the XGBoost model's predictions on the testing set. The surrogate models `surrogate_model_rf` and `surrogate_model_xgboost` are both relatively effective in capturing the variance of the original model, as the count R2 for `surrogate_model_rf` is 0.8472455167075216 on the testing set, while the count R2 for `surrogate_model_xgboost` is 0.8440201264352987 on the testing set.

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.

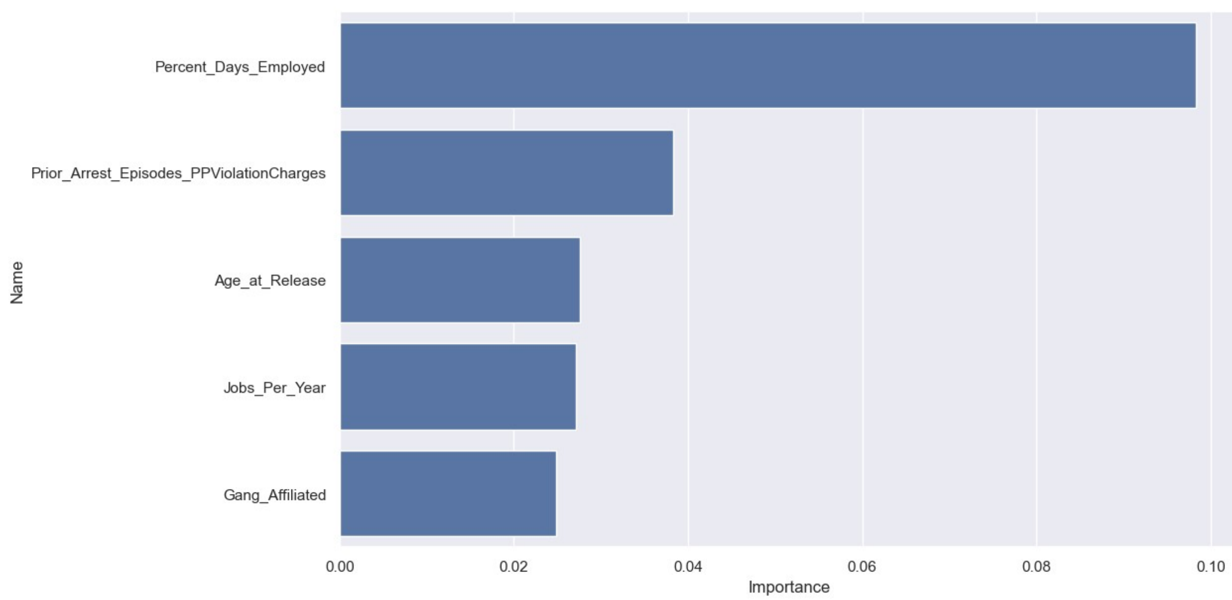
Luckily for us, Permutation Feature Importance already exists as a function in Scikit-Learn! All you have to do it is looking at the documentation to learn how it works, and apply it to the 3 non-inherently interpretable models of this exercise. Let's start with Random Forest.

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_train, y_train, n_repeats=10,
random_state=123)
# out
```

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.

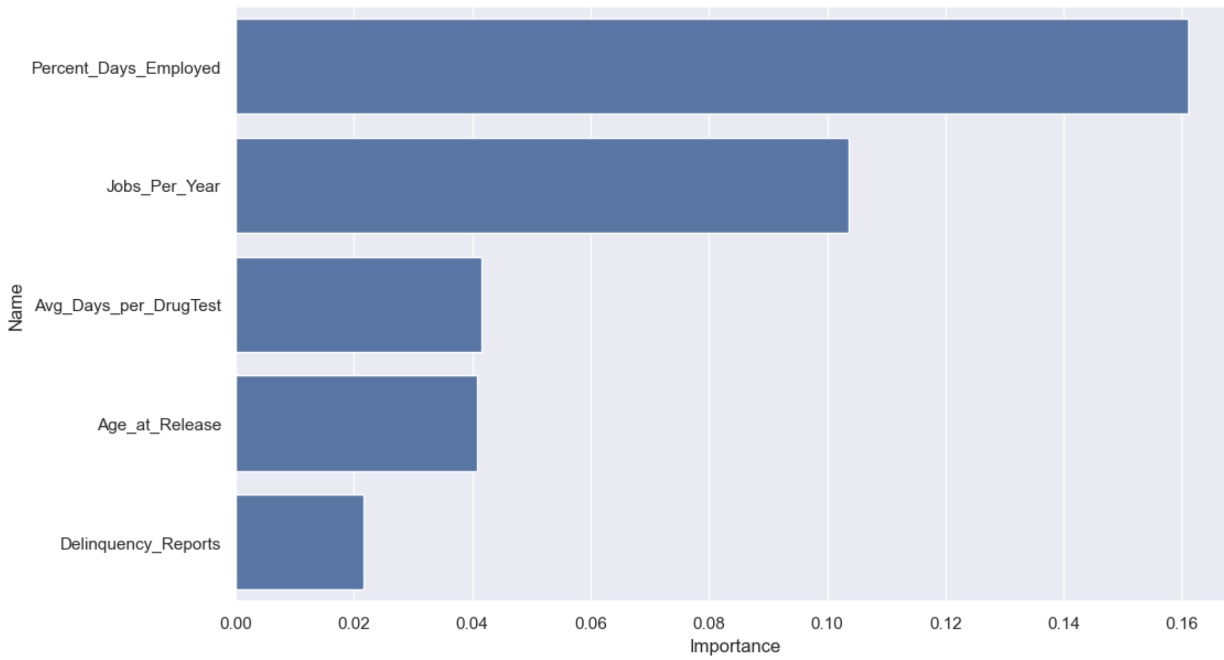
Hint: this is a more complex model; if you find that this task is taking too long, you may consider reducing the number of permutations using the parameter `n_repeats`. Be aware that this produces more variable results.

XGBoost Model:

```
out_2 = permutation_importance(xgboost_model, X_train, y_train,
n_repeats=10, random_state=123)
```

```
result_2 = pd.DataFrame({"Name": X_test.columns, "Importance":
out_2["importances_mean"], "STD": out_2["importances_std"]})
result_2 = result_2.sort_values(by=['Importance'], ascending=False)

sns.set(rc={'figure.figsize':(11.7,7)})
sns.barplot(data=result_2[: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?

`Percent_Days_Employed`, `Age_at_Release` and `Jobs_Per_Year` are features that are highly important to both `rf_model` and `xgboost_model`. `Percent_Days_Employed` is the most important feature in both models, which is similar to how their corresponding surrogate models and `logreg_model` provide a very large negative coefficient for `Percent_Days_Employed`. `Age_at_Release` has more importance in `xgboost_model` than in `rf_model`, and both of their corresponding surrogate models along with `logreg_model` often provide large coefficients for specific levels of `Age_at_Release`, with `pipeline-2__Age_at_Release_48 or older` in particular consistently having a large negative coefficient. `Jobs_Per_Year` has significantly greater importance in `xgboost_model` than in `rf_model`, yet the surrogate model for `xgboost_model` does not have the feature as among its greatest coefficients, while the feature has a large coefficient for `logreg_model` and the surrogate model of `rf_model`. The feature importance of `Prior_Arrest_Episodes_PPViolationCharges` in `rf_model` corresponds to the large negative coefficient of `pipeline-2__Prior_Arrest_Episodes_PPViolationCharges_0` and the large positive coefficient of `pipeline-2__Prior_Arrest_Episodes_PPViolationCharges_5 or more` from `surrogate_model_rf`, which are among the extreme levels for `Prior_Arrest_Episodes_PPViolationCharges`. The feature importance of `Prior_Arrest_Episodes_PPViolationCharges` in `rf_model` corresponds to the large positive coefficient of `pipeline-2__Gang_Affiliated_True` from `surrogate_model_rf`. The feature importance of `Avg_Days_per_DrugTest` in `xgboost_model` does not correspond to the features with particularly large coefficient values provided by `surrogate_model_xgboost`. The feature importance of `Delinquency_Reports` corresponds to the large negative coefficient for `pipeline-2__Delinquency_Reports_4 or more` and the large positive coefficient for `pipeline-2__Delinquency_Reports_1` from `surrogate_model_xgboost`, which are among the extreme levels for `Delinquency_Reports`.

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:

```
# !pip install shap
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,
)

np.random.seed(1234)
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(X_test_enc.iloc[ind])  # explaining
predictions for 1000 random samples
rf_shap_values = rf_explainer.shap_values(X_test_enc.iloc[ind])

display(rf_shap_values[:,:,1])
# display(rf_shap_values.values)
# display(rf_shap_values_plot)

array([[ 6.35987948e-03, -2.10280953e-02, -1.14059370e-02, ...,
         -1.29857993e-03,  2.81089984e-05, -8.04362727e-04],
       [-7.98872173e-04, -3.39887698e-02, -6.53120312e-03, ...,
         -1.45430959e-03,  3.23441046e-05, -7.81584252e-04],
       [ 5.07039050e-03,  1.99693455e-02, -1.91157485e-02, ...,
          4.19122486e-03,  2.62278452e-04, -2.05489404e-04],
       ...,
       [ 2.76394898e-03,  6.00263132e-03,  3.25624454e-03, ...,
         -3.48703945e-04,  7.05770915e-04, -3.99763100e-04],
       [-4.42869122e-03, -3.39369379e-02, -1.41858770e-02, ...,
         -7.69076182e-04, -1.86780577e-04,  4.03759553e-04],
       [-3.94334448e-03,  8.21011060e-03, -4.81923980e-03, ...,
         -7.63568655e-04,  3.85609483e-04,  3.01598369e-03]])
```

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

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

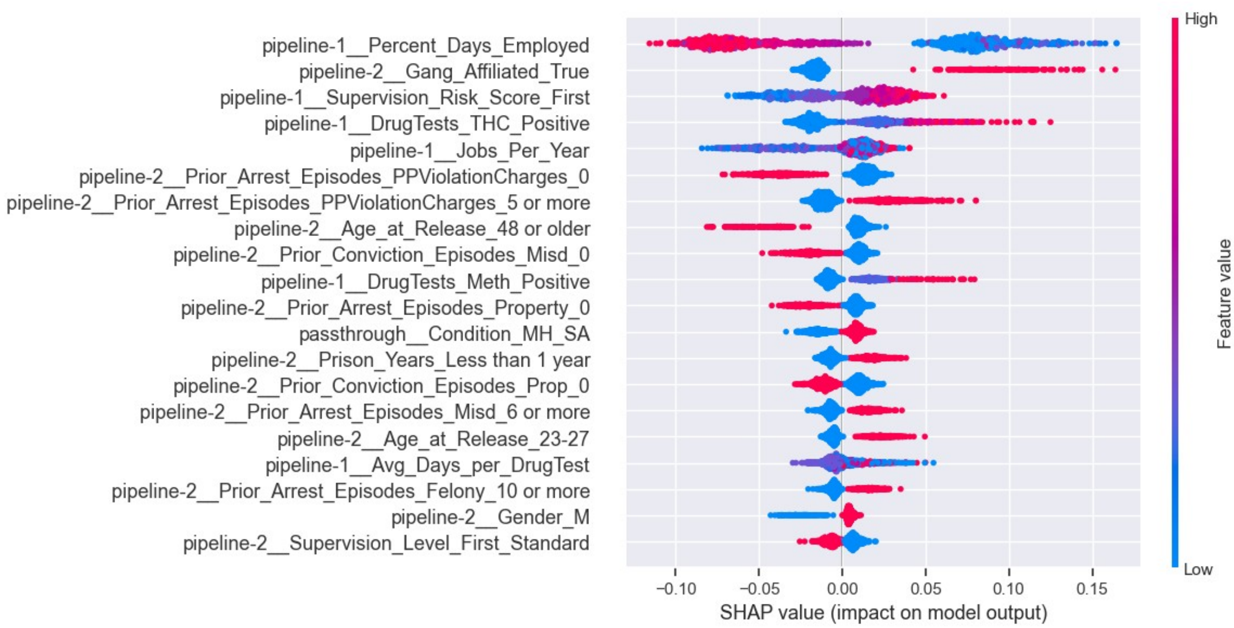
```
# values = np.abs(rf_shap_values.values).mean(0)
values = np.abs(rf_shap_values[:,:,1]).mean(0)
# values
```

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

	SHAP
pipeline-1__Percent_Days_Employed	0.073569
pipeline-2__Gang_Affiliated_True	0.028402
pipeline-1__Supervision_Risk_Score_First	0.025395
pipeline-1__DrugTests_THC_Positive	0.025105
pipeline-1__Jobs_Per_Year	0.020859
pipeline-2__Prior_Arrest_Episodes_PPViolationCh...	0.020744
pipeline-2__Prior_Arrest_Episodes_PPViolationCh...	0.016905
pipeline-2__Age_at_Release_48 or older	0.015044
pipeline-2__Prior_Conviction_Episodes_Misd_0	0.013325
pipeline-1__DrugTests_Meth_Positive	0.012363

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])
# shap_figure = shap.summary_plot(rf_shap_values[1],
X_test_enc.iloc[ind], plot_size=[12,6])
# shap_figure = shap.summary_plot(rf_shap_values.values,
X_test_enc.iloc[ind], plot_size=[12,6])
# shap.plots.beeswarm(rf_shap_values[:,:,1])
```



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.

The beeswarm plot for `rf_model` provides distinct groupings or direct scalings of SHAP values for most of the displayed features based on their values, with the exception of `pipeline-1__Supervision_Risk_Score_First`, `pipeline-1__DrugTests_THC_Positive`, `pipeline-1__Jobs_Per_Year`, `pipeline-1__DrugTests_Meth_Positive` and `pipeline-1__Avg_Days_per_DrugTest`. `pipeline-1__Percent_Days_Employed` and `pipeline-2__Gang_Affiliated_True` have very extreme ranges of SHAP values, implying very strong contributions on the model prediction, which is similar to the results from the corresponding surrogate model and permutation importance graph. `pipeline-1__Supervision_Risk_Score_First`, `pipeline-1__DrugTests_THC_Positive` and `pipeline-1__Jobs_Per_Year` provide relatively extreme SHAP value ranges, despite not being among the results from the corresponding surrogate model and permutation importance graph, which is potentially explained by the lack of fully distinct groupings between feature values.

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
# xgboost_shap_values

xgboost_values = np.abs(xgboost_shap_values).mean(0)
# xgboost_values

pd.DataFrame(data=xgboost_values, index=feature_names, columns=["SHAP"]).sort_values(
    by="SHAP", ascending=False
)[:10]
```

	SHAP
pipeline-1__Percent_Days_Employed	0.762783
pipeline-1__Jobs_Per_Year	0.481763
pipeline-1__Avg_Days_per_DrugTest	0.198515
pipeline-2__Gang_Affiliated_True	0.189281
passthrough__Condition_MH_SA	0.145446
pipeline-1__DrugTests_THC_Positive	0.136942
pipeline-1__DrugTests_Meth_Positive	0.136918
pipeline-2__Age_at_Release_48 or older	0.120194
pipeline-2__Gender_M	0.110206
pipeline-1__Supervision_Risk_Score_First	0.108372

```
shap_figure = shap.summary_plot(xgboost_shap_values,
X_test_enc.iloc[ind], plot_size=[12,6])
```

The beeswarm plot for `xgboost_model` provides distinct groupings or direct scalings of SHAP values for most of the displayed features based on their values, with the exception of `pipeline-1__Supervision_Risk_Score_First`, `pipeline-1__DrugTests_THC_Positive`, `pipeline-1__Jobs_Per_Year`, `pipeline-1__DrugTests_Meth_Positive` and `pipeline-1__Avg_Days_per_DrugTest`. `pipeline-1__Percent_Days_Employed` and `pipeline-2__Gang_Affiliated_True` have very extreme ranges of SHAP values, implying very strong contributions on the model prediction, which is similar to the results from the corresponding surrogate model and permutation importance graph. `pipeline-1__Supervision_Risk_Score_First` and `pipeline-1__DrugTests_THC_Positive` provide relatively extreme SHAP value ranges, despite not being among the results from the corresponding surrogate model and permutation importance graph, which is potentially explained by the lack of fully distinct groupings between feature values.

Question 14: Explaining individual predictions using SHAP

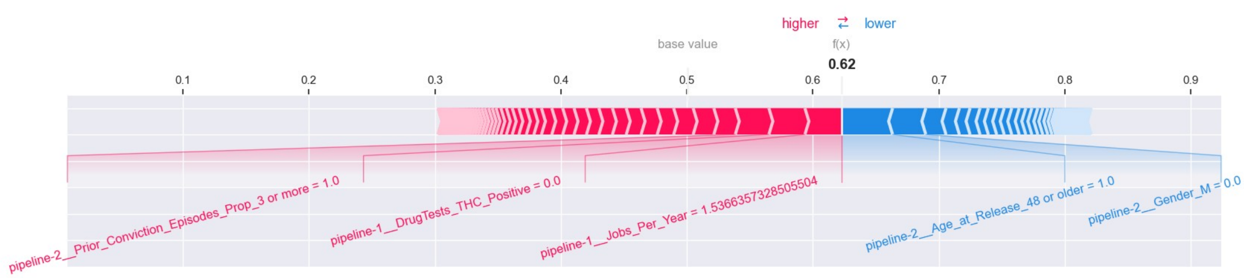
Another powerful feature of SHAP is that it allows us to explain the impact of each feature on individual predictions. For example, we will be able to explain how the prediction for our hard sample was generated. Let's start by looking at the prediction for this sample given by the random forest model.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

```
# Your answer here: The classification is incorrect
display(y_test[106:107]) # False
display(rf_model.predict(hard_sample)) # True
```

```
106    False
Name: Recidivism_Within_3years, dtype: bool

array([ True])
```

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

```
# shap.force_plot(
#     rf_explainer.expected_value[1],
#     rf_shap_values[1][-1],
#     X_test_enc.iloc[ind[-1]],
#     matplotlib=True,
# )
display(rf_explainer.expected_value[1]) # 0.5004571709571111

shap.force_plot(
    rf_explainer.expected_value[1],
    rf_shap_values[:,:,1][-1],
    X_test_enc.iloc[ind[-1]],
    matplotlib=True,
    figsize=(20, 3),
    text_rotation=15,
)

0.5004571709571111
```



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?

Random Forest Model: `pipeline-1__Jobs_Per_Year = 1.5366` (4 d.p.) provides the greatest contribution to the prediction of `Recidivism_Within_3years = True`, while `pipeline-2__Age_at_Release_48 or older = 1.0` provides the greatest counter to the prediction. Each side has relatively equal numbers of significant contributions, the total contributions from the positive direction appears slightly greater than that from the negative direction, and the prediction value of 0.62 does not significantly deviate from the base value of 0.5004571709571111. With `rf_model.predict(hard_sample) = True` matching with the SHAP prediction value of 0.62, the prediction was not difficult.

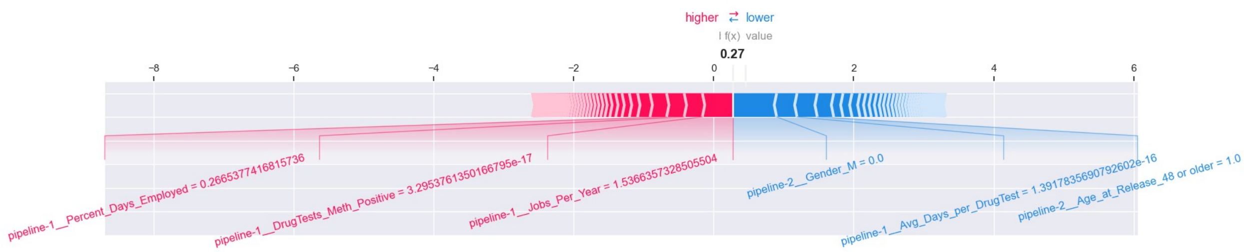
Finally, **repeat the analysis and comment on the results of the individual predictions made on the hard sample by XGBoost and Decision Tree** (since we were not able to do the latter earlier).

```
# Your answer here
display(y_test[106:107]) # False
display(xgboost_model.predict(hard_sample)) # True
display(xgboost_explainer.expected_value) # 0.45338702

shap.force_plot(
    xgboost_explainer.expected_value,
    xgboost_shap_values[-1],
    X_test_enc.iloc[ind[-1]],
    matplotlib=True,
    figsize=(20, 3),
    text_rotation=15,
)
```

```
106    False
Name: Recidivism_Within_3years, dtype: bool
```

```
array([1])
```

```
0.45338702
```



XGBoost Model: `pipeline-1__Jobs_Per_Year = 1.5366` (4 d.p.) provides the greatest contribution to the prediction of `Recidivism_Within_3years = True`, while `pipeline-2__Gender_M = 0.0` provides the greatest counter to the prediction. This was a difficult prediction, as each side has relatively equal numbers of significant contributions, the total contributions from each direction appears to be relatively equal, and the prediction value of 0.27 is significantly less than the base value of 0.45338702. Additionally, the prediction `xgboost_model.predict(hard_sample)` is `1`, and assuming this corresponds to `Recidivism_Within_3years = True`, this causes `xgboost_model.predict(hard_sample) = 1` to conflict with the SHAP prediction value of 0.27.

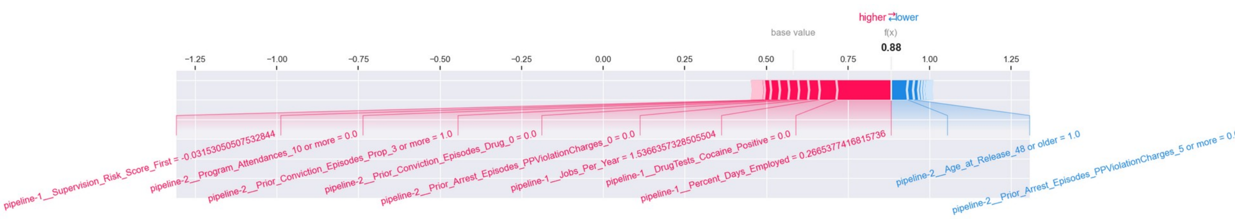
```
display(y_test[106:107]) # False
display(tree_model.predict(hard_sample)) # True

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]) # explaining predictions for 1000 random samples
```

```
display(tree_explainer.expected_value[1]) # 0.5820614908206149
# tree_shap_values[:,:,1]
shap.force_plot(
    tree_explainer.expected_value[1],
    tree_shap_values[:,:,1][-1],
    X_test_enc.iloc[ind[-1]],
    matplotlib=True,
    figsize=(20, 3),
    text_rotation=15,
)
```

```
106    False
Name: Recidivism_Within_3years, dtype: bool
```

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

```
0.5820614908206149
```



Decison Tree Model: `pipeline-1__Percent_Days_Employed = 0.2665` (4 d.p.) provides the greatest contribution to the prediction of `Recidivism_Within_3years = True`, while `pipeline-2__Age_at_Release_48 or older = 1.0` provides the greatest counter to the prediction. This was not a difficult prediction, as the majority of significant contributions were towards the positive direction, the total contributions from the positive direction appears greater than that from the negative direction, and the prediction value of 0.88 is significantly greater than the base value of 0.5820614908206149, along with corresponding to `tree_model.predict(hard_sample) = True`.

Part 5: Final Evaluation:

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).

All of the models provide relatively similar values of `f1 score`, which is arguably more important as a performance metric due to the focus on the positive class of `Recidivism_Within_3years`. Given that the application of the model is for the the identification of across populations of race, measurement bias will likely occur to the detriment of black defendants. As such, the fairness metrics `FDRD_adfr` and `FPRD_adfr` should be minimised to reduce both the likelihood of black defendants being predicted as true in general and the likelihood of black defendants being predicted as guilty when they are actually innocent. For this application, `xgboost_model` should be employed, as it provides the second greatest testing `f1 score`, though there is a significant risk of overfit on the original dataset, and it provides the lowest `FDRD_adfr` and `FPRD_adfr` values among the models, indicating that it is relatively the most fair. The model also consistently has the features `Percent_Days_Employed`, `Jobs_Per_Year`, `Age_at_Release`, `Gang_Affiliated`, `Condition_MH_SA` and `Gender_M` with particularly high importance, though there is some ambiguity with several features, such as `Avg_Days_per_DrugTest`, and the SHAP beeswarm plot shows relatively less features involving prior arrests, convictions, and revocations.

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 worked on the assignment separately, each taking turns to answer all parts and modifying the responses down the line.

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.

- Jingyuan's response: Used ChatGPT to help with fixing errors in environment and debugging the "import" statements. I also used ChatGPT to help with the `ConfusionMatrixDisplay` and `confusion_matrix` functions.
- Nicholas' response: Used ChatGPT to help with fixing errors in environment and understanding how surrogate model could be used to interpret feature influence in non-interpretable models in Q11.

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!

- Jingyuan's response: I am still struggling with the surrogate model and its interpretation. I spent a lot of time trying to understand how to evaluate non-inherently interpretable models but are still somehow confused.
- Nicholas' response: Interpreting feature influence of non-interpretable models through surrogate model, modifying provided code to allow SHAP plots in Q13 to function properly, interpreting SHAP plots in general.