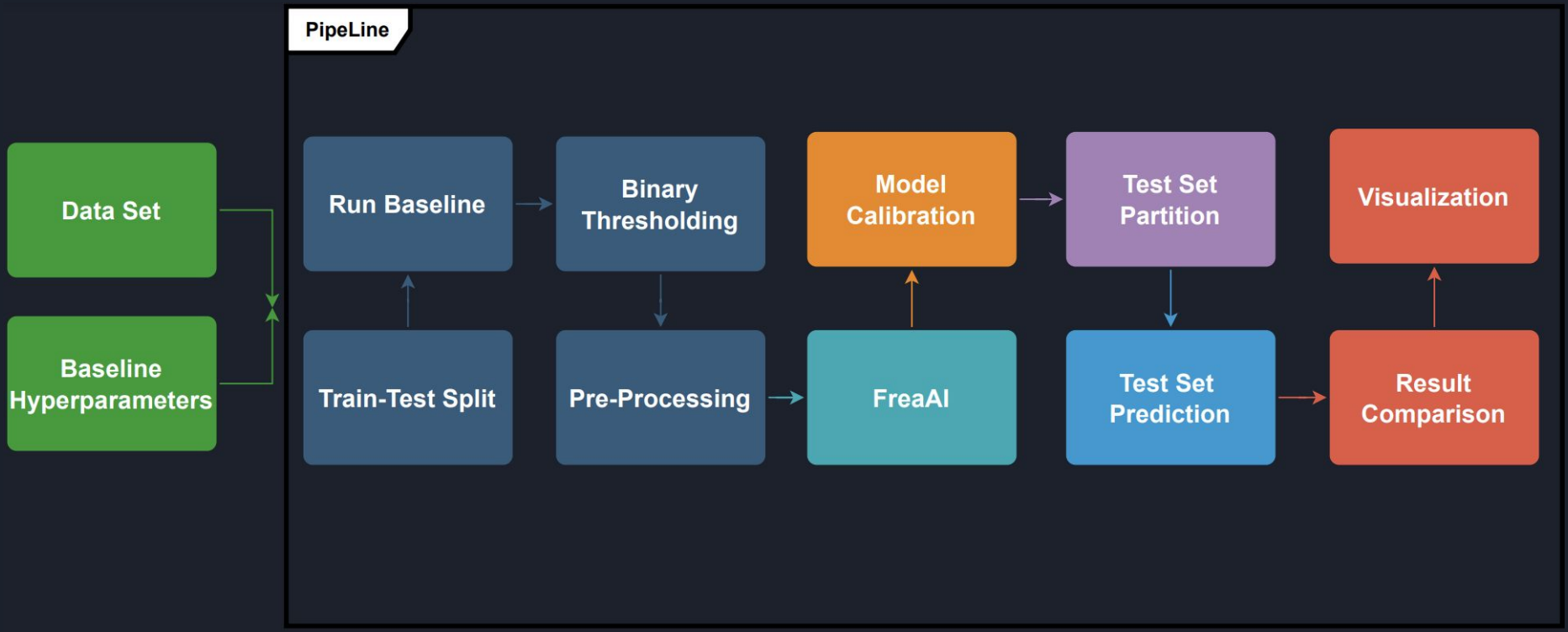
Decision Tree Boost: A Machine Learning Pipeline for Improved XGBoost Modeling and Prediction

Boaz Zimbler, Adam Vinestock, Itay Shapira, Omri Newman
Reichman University
January 2, 2023

Table of Contents

Pipeline Architecture

GitHub Repo



Pre-Processing

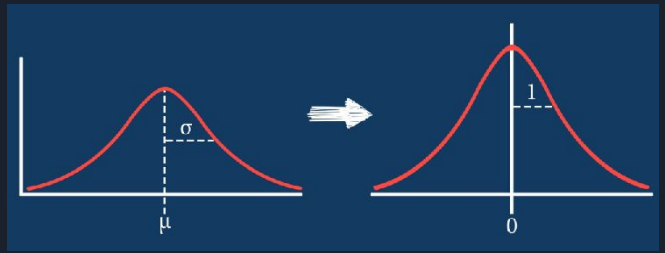
```
def create_pre_process_pipeline(self):
  num_pipeline = Pipeline([('std_scaler', StandardScaler())])
  preprocessor = ColumnTransformer(
        transformers=[
          ('numeric', num_pipeline, self.numeric_features),
          ('categorical', OrdinalEncoder(), self.categorical_features)])
  self.pre_process_pipeline = preprocessor
```

Categorical

VehGas	Count
Regular	345877
Diesel	332136

Area	Count
C	191880
D	151596
E	137167
A	103957
B	75459
F	17954

Numerical



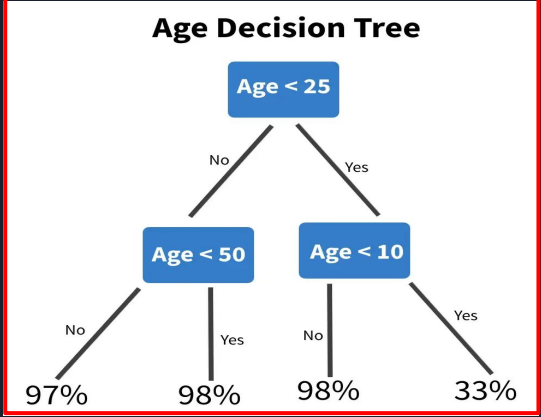
FreaAI

What

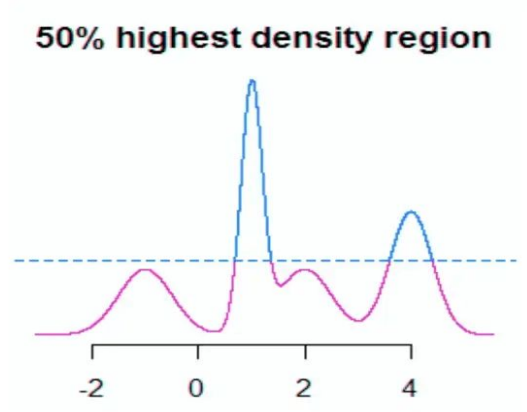
Automated Extraction of Data Slices to Test ML Models
Identifying **Weak Data** which **underperforms in the model**

How

Decision Trees



HDR



FreaAI

Why

Business **Decision Making**

"**Focusing**" on specific market

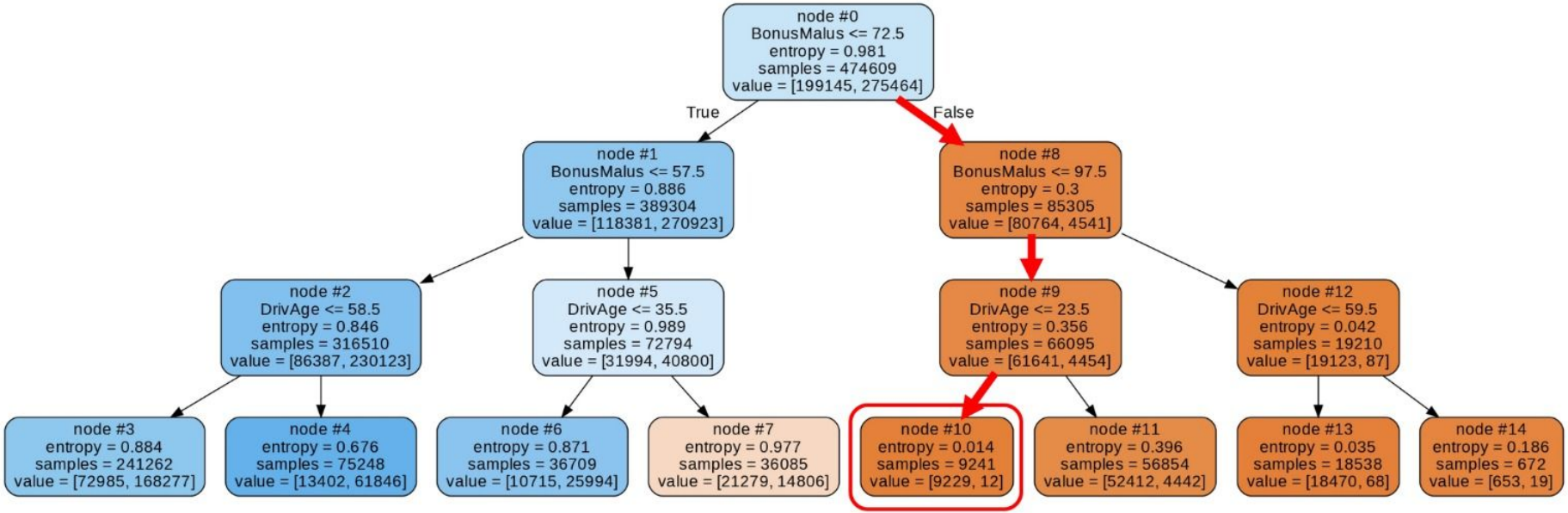
Rely on model only in **specific** market

Improving model by dealing with weak data

Getting more Weak Data to train on

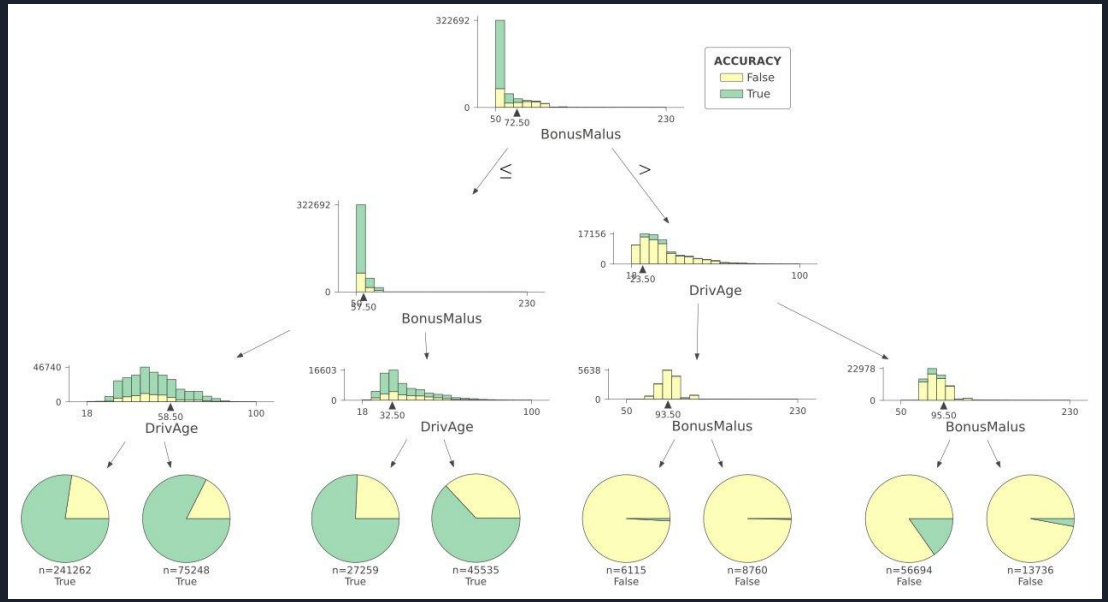
Splitting models to reduce generalization error

FreaAI


node #0
BonusMalus <= 72.5
entropy = 0.981
samples = 474609
value = [199145, 275464]
True
False
node #1
BonusMalus <= 57.5
entropy = 0.886
samples = 389304
value = [118381, 270923]
node #8
BonusMalus <= 97.5
entropy = 0.3
samples = 85305
value = [80764, 4541]
node #2
DrivAge <= 58.5
entropy = 0.846
samples = 316510
value = [86387, 230123]
node #5
DrivAge <= 35.5
entropy = 0.989
samples = 72794
value = [31994, 40800]
node #9
DrivAge <= 23.5
entropy = 0.356
samples = 66095
value = [61641, 4454]
node #12
DrivAge <= 59.5
entropy = 0.042
samples = 19210
value = [19123, 87]
node #3
entropy = 0.884
samples = 241262
value = [72985, 168277]
node #4
entropy = 0.676
samples = 75248
value = [13402, 61846]
node #6
entropy = 0.871
samples = 36709
value = [10715, 25994]
node #7
entropy = 0.977
samples = 36085
value = [21279, 14806]
node #10
entropy = 0.014
samples = 9241
value = [9229, 12]
node #11
entropy = 0.396
samples = 56854
value = [52412, 4442]
node #13
entropy = 0.035
samples = 18538
value = [18470, 68]
node #14
entropy = 0.186
samples = 672
value = [653, 19]

Model Calibration

- Based on FreaAI we identify ‘Low quality’ subsets of data
- Extract Low quality data from the training set splitting it into two - Low and High quality data
- Train two new XGBoost models calibrated to data sets



Test Set Partition and Prediction

Partition test set into two
High quality data | Low quality data
as determined by FreaAI decision trees

Predict on partitioned test sets
using corresponding models from
Calibration step

Additional Optimization Approaches

‘High quality’ vs ‘Low quality’ data performance

Creating more robust data partition - low quality data detection using ensemble of trees

```
[31] # Extracting more robust poor quality data

     def mask_repeated_elemer
         unique_elements, cou
         repeated_elements =
         return repeated_elem

     def flagged_data(df, out
         poor_data = np.empty
         n_trees = len(output
         for i in range(n_tre
           features = output.
           if isinstance(feat
             pass
           else:
               features = [fe
           tree_model = outpu
           leaf = output.iloc
           leaf_indices = tre
           bad_indices = np.a
           poor_data = np.cor
         flagged_data = mask_
         print(f"Num of min a
         print(f"Total size c
         return flagged_data

     def classify_samples(X,
         low_quality_indices_
         for i in range(trees
           features = trees.i
           if isinstance(feat
             pass
           else:
             features = [features]
```

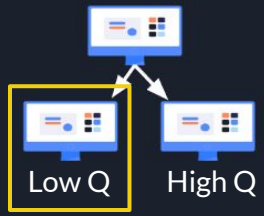
```
[108] # Preproccessing the data
      features_flagged = np.concatenate((french_df.columns.["FLAGGED"]))
      mask = frenc
      flags = np.w
      X_flagged =
      df_flagged =

      df_flagged['
      df_flagged['
      df_flagged.h
```

	Area	Vel								
0	0.0									
1	0.0									
2	1.0	6.0	2.0	52.0	50.0	0.0	1.0	54.0	22.0	0.000000	0.0
3	1.0	7.0	0.0	46.0	50.0	0.0	1.0	76.0	72.0	0.000000	0.0
4	1.0	7.0	0.0	46.0	50.0	0.0	1.0	76.0	72.0	1.190476	0.0

```
---------- Test data summary using threshold of 4 tree's ----------
Baseline
 predictions within threshold: 64.02% , MSE: 31.215869537406565
--------------------------------------------------------------------------------
Low quality data
 predictions within threshold: 19.01% , MSE: 67.26537746807695
--------------------------------------------------------------------------------
High quality data
 predictions within threshold: 76.07% , MSE: 9.933800932185752
--------------------------------------------------------------------------------
Weighted sum
 predictions within threshold: 65.91% , MSE: 31.422290499466996
```

Low-quality Data Generation

Splitting models to reduce generalization error



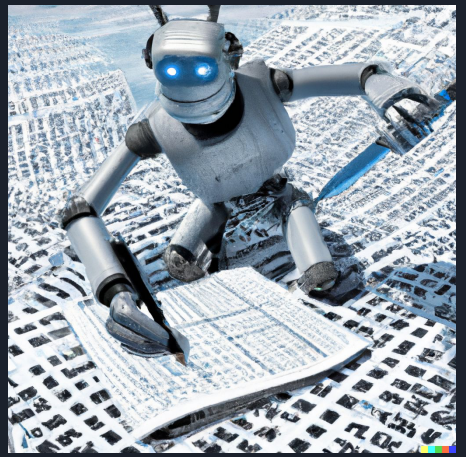

```
------------------- Test data summary using CWGAN-GP -----------------
Baseline
 predictions within threshold: 64.02% , MSE: 31.215869537406565
--------------------------------------------------------------------------
Low quality data
 predictions within threshold: 8.02% , MSE: 46.486560008961625
--------------------------------------------------------------------------
High quality data
 predictions within threshold: 76.07% , MSE: 27.923652375874138
--------------------------------------------------------------------------
Weighted sum
 predictions within threshold: 64.04% , MSE: 31.228953100811516
```

Imp

De

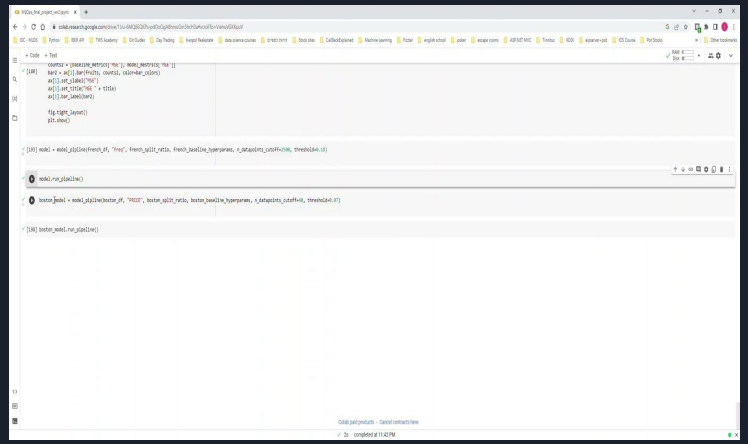
Oversampling using Gaussian Copulas

CWGAN-GP data generation using YData

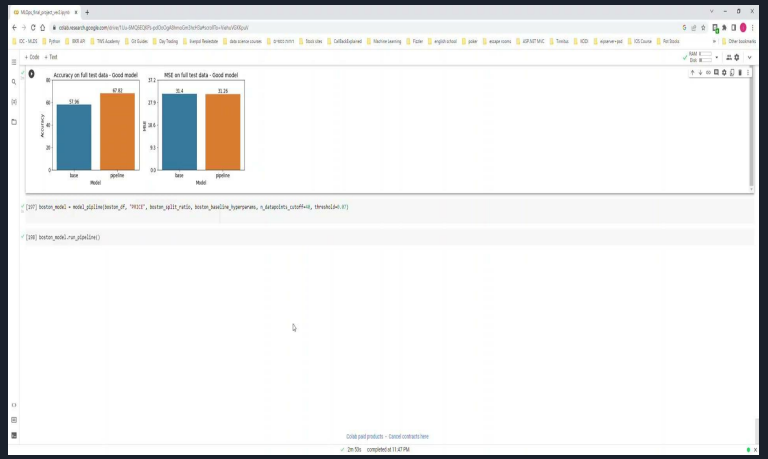


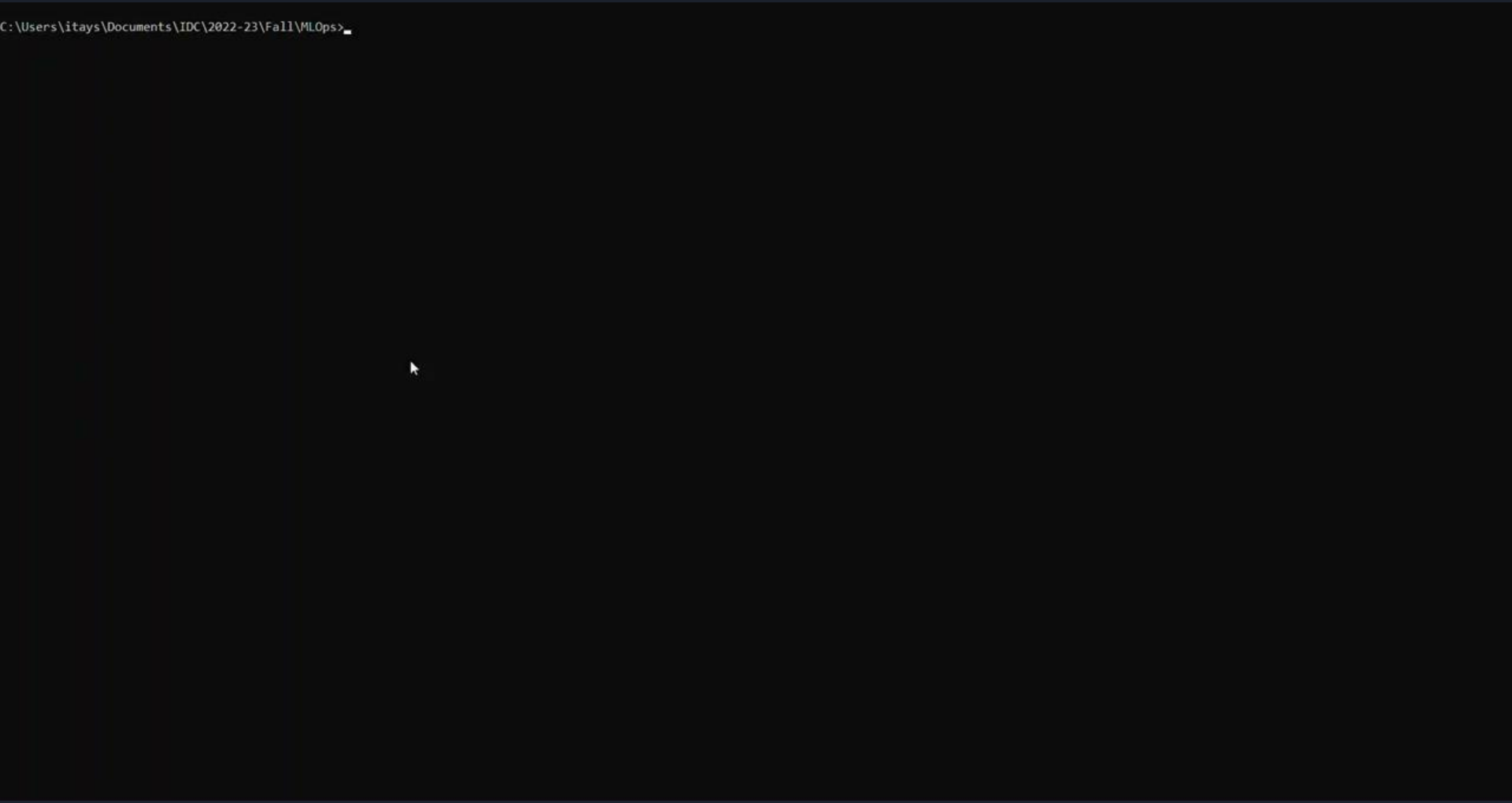
Code Demo

French

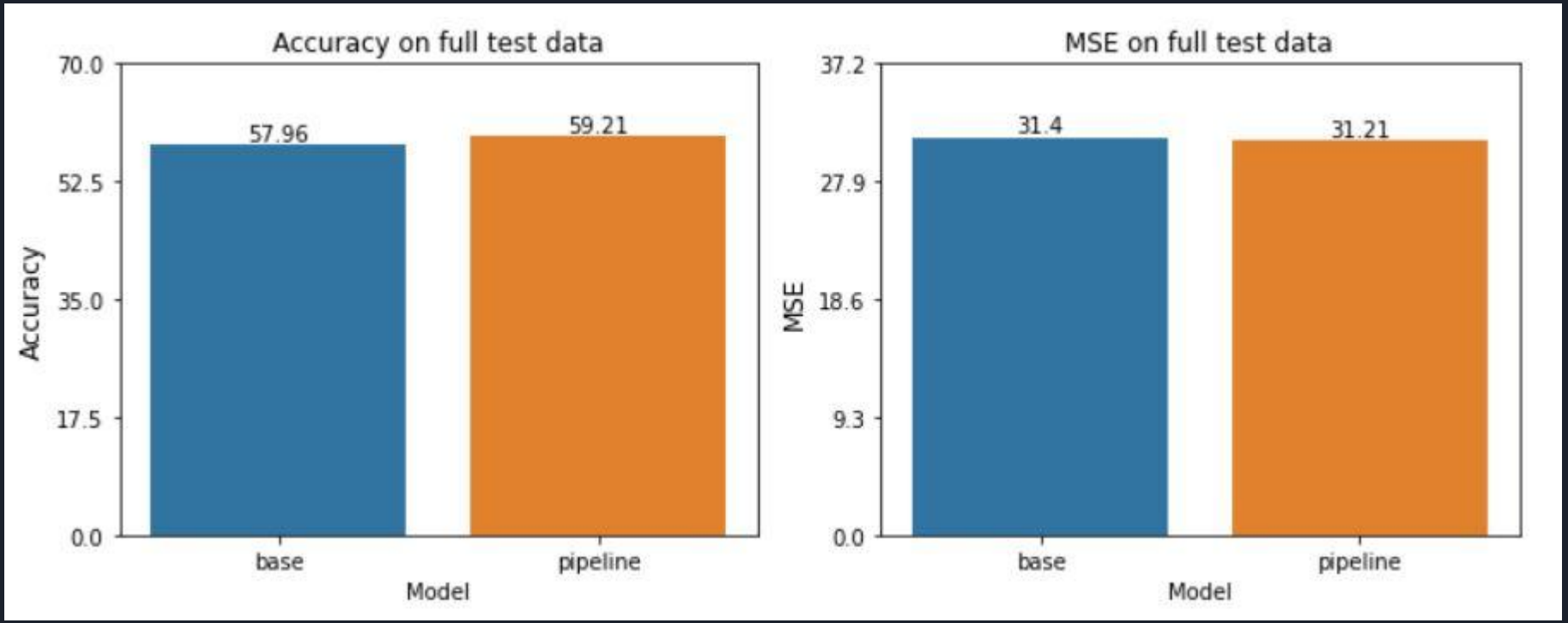


Boston

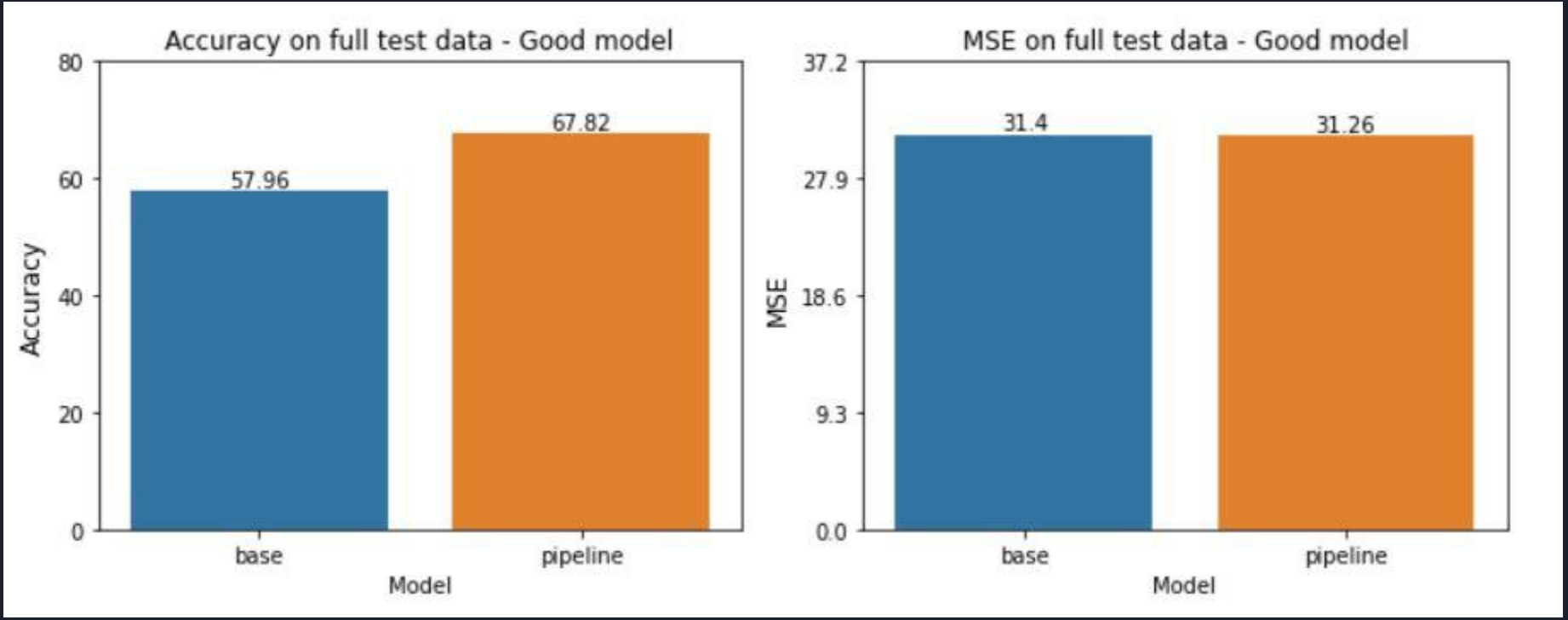



C:\Users\itays\Documents\IDC\2022-23\Fall\MLOps>

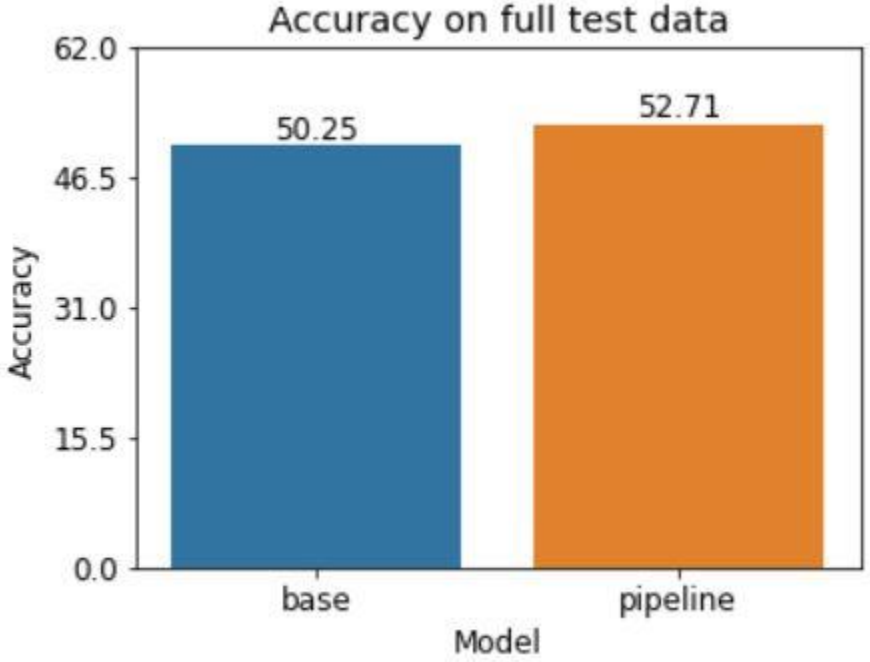
Results and Comparison (French)

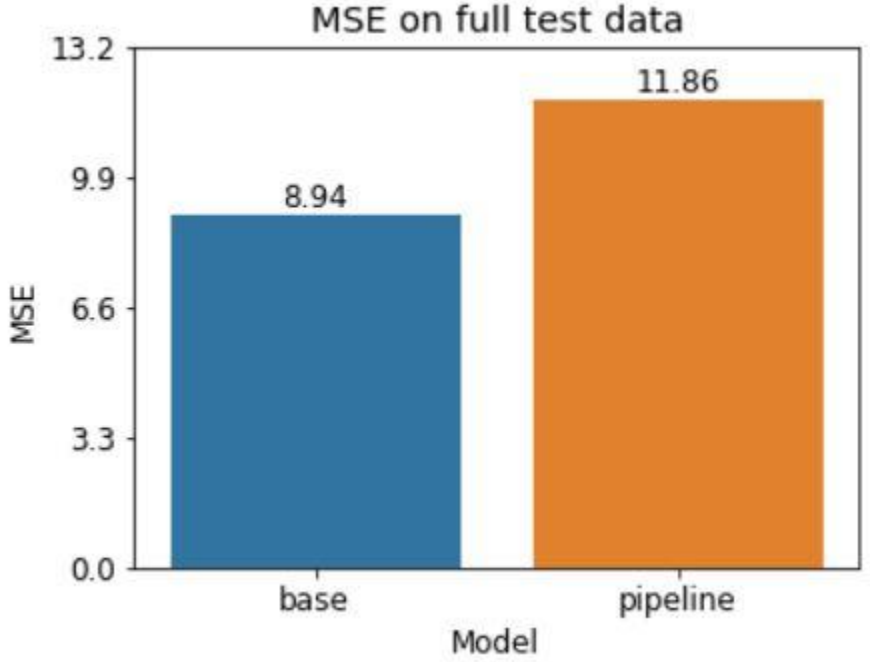


Results and Comparison (French)

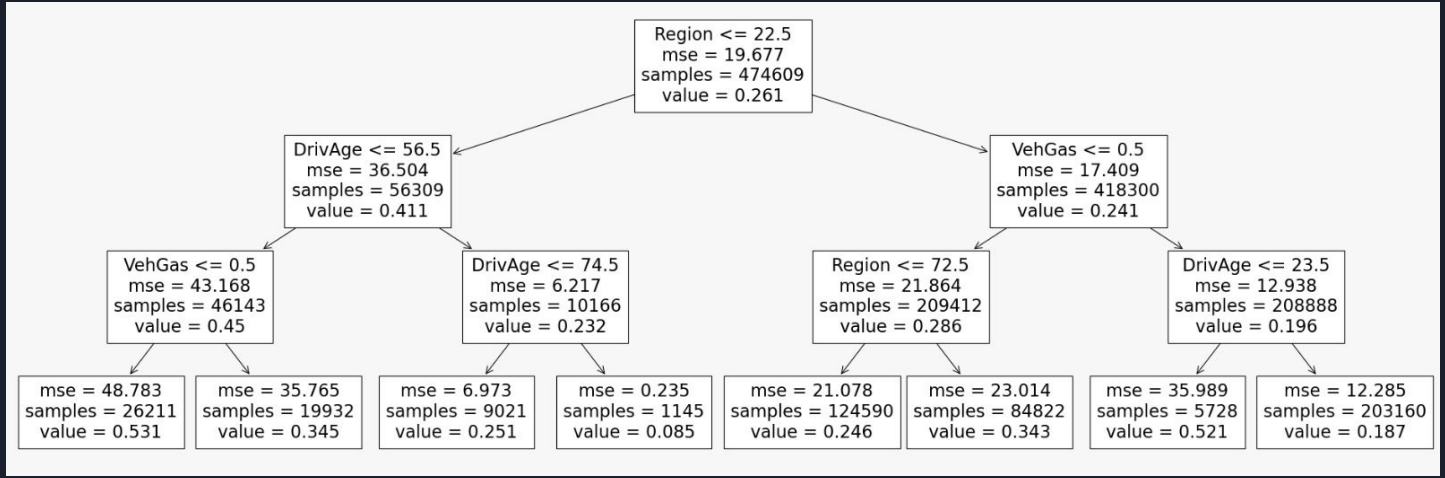


Results and Comparison (Boston)





Next Steps



- More Data Collection
- Implementing HPD on top of Decision Trees
- Further explore FreaAI using Decision Tree Regressors

Q&A?

Thanks!