ISE 625 Project Progress

Stable decision trees for suicide experience prediction

Adhithya Bhaskar, Michelle Gelman

Problem Context and Background

- **Aim:** Predict suicidal experiences among youth experiencing homelessness (YEH)
- The provided decision tree model is unstable to change in train-test splits
- Can we find a robust model invariant to shifts in distributions that will procude the same best features indicative of suicide ideaiton and attempts?

Dataset Considerations

- Missing data
  - 584, 587 samples remaining for each prediction model from initial listwise deletion method form oriignal 940 total samples
  - 4% of data set mising for `suicideidea` and `suicideattempt` (36 and 40 samples respectively)
- Imbalanced classes
  - 83% labeled 2, 16% labeled 1 for suicideidea class
  - 88% labeled 0, 11% labeled 1 for suicideattempt class

Stable Decision Trees

- Bertsimas et al. (2023) proposes a method to create stable decision trees
- 1 of 6 datasets used is publicly available - Breast Cancer dataset (UCI Machine Learning Repository)
- Used to test and compare our implementation
- With satisfactory results, we will apply our implementation to the suicide dataset

Proposed Plan - 1. Understand the instability of provided DT

- Given model exists as 2 python files (for `suicidea` and `suicattemp`)
- Create simple example to deterministically try various splits
- Empirically measure the difference in predicted splits

Proposed Plan - 2. Implement a stable DT (Bertsimas et al. 2023)

1. Train initial set **(T0)** of decision trees on a subset of the data and a second set **(T)** on the full dataset
2. Compute **average distance** of each tree in **T** to **T0** *(found code for this)*
3. Compute performance metrics (AUC) of trees on validation/test set
4. For the trees in **T** we select the Pareto optimal trees by optimizing for predictive performance and distance to **T0**

Proposed Plan - 3. Measuring effectiveness of proposed model

1. Evaluate performance of provided DT using the stability experiment we define in step 1
2. Evaluate performance of the stable tree using the same experiment handler
3. Define and compare the models using metrics for assessing stability over various splits

Key optimization algorithms to implement

- ~~Distance between two trees~~

$$d(\mathcal{T}_1, \mathcal{T}_2) \;=\; \min_{\{x\}} \sum_{p \in \mathcal{P}(\mathcal{T}_1)} \sum_{q \in \mathcal{P}(\mathcal{T}_2)} d(p,q)\, x_{p,q} + \sum_{p \in \mathcal{P}(\mathcal{T}_1)} w(p)\, x_p$$

- Pareto optimal tree

$$\mathbb{T}^\star = \operatorname{argmax} f(d_b, \alpha_b)$$

Implementation progress

- Jupyter Notebook
- Refactor provided code to create a pipeline for the stable decision tree

Provided code (1/2)

```
df = origindf[['age', 'gender', 'sexori', 'raceall', 'trauma_sum', \
               'cesd_score', 'harddrug_life','school','degree','job',\
               'sex', 'concurrent', 'exchange', 'children',\
               'weapon','fight', 'fighthurt', 'ipv', 'ptsd_score', 'alcfirst', \
                .
                .
                .
               labelstr]].copy()
logging.info("Cleaning sub tree from NaN")
dfn = df.dropna().copy()
train_test_cutoff=int(round(dfn.shape[0]*.75))
```

Provided code (2/2)

```
dfm = dfn.values
X_train, y_train = dfm[0:train_test_cutoff,:-1], dfm[0:train_test_cutoff,-1]
X_test, y_test = dfm[train_test_cutoff:,:-1], dfm[train_test_cutoff:,-1]
y = dfm[:,-1]

cw = compute_class_weight(class_weight='balanced', classes=np.unique(y), y=y)
cwt={0:cw[0], 1:cw[1]}

clf = DecisionTreeClassifier(criterion='gini',min_samples_leaf=10, min_samples_split=20, max_
clf.fit(X_train, y_train)
```

Project Outcomes

- A robust, stable decision tree model that minimizes the variability in tree structure due to random train-test splits
- Empirical evidence supporting the stability of the model through consistent feature selection and comparable performance metrics
- **Impact:** Better interpretability of decision trees to predict suicide risk among YEHs