Machine Learning Applications

on

Survey Data Prediction

Target Prediction Variable: Childhood Experience of Violence

A comparative study of traditional and machine learning approaches for prediction analysis on KONDA Research and Consultancy survey data from July 2023.

By: Orhan Akpınar

Department: Koç University, Computational Social Sciences MA

Date: 20 January 2024

Introduction

Social sciences are experiencing a new era with the use of computational methods. Beforehand, theories were not so easy to test, and therefore, models relied more on the theoretical assumptions and approach of researchers. Nowadays, computational methods allow for in-depth testing of traditional models. Furthermore, they provide novel approaches to tackle social problems. In this regard, I wanted to utilize both sides of the benefits of computational methods.

Briefly, my analysis focuses on two parts. In the first part, I will compare the traditional approach with a machine learning model with hand-made variable sets versus full set variables. In the second part, I will compare different machine learning classification algorithms for prediction score analysis.

Inevitably, machine learning models are superior to hand-tailored methods of testing. Traditionally, researchers look for the explainability of their design. Therefore, using too many variables (features in ML language) would not help to increase explainability. As a result, it is not usual to incorporate every part of a survey for assessing outcomes. In the analysis, researchers should drop extra or unnecessary parts that don't regard their question. It is the opposite of the machine learning approach.

In the machine learning model, the more the merrier. Because ML relies on heavy use of data, in return we expect every bit of data to be meaningful for achieving results. What is also different is that ML models can compute many equations and combinations simultaneously, and therefore their method would be superior to hand-made equations. Nowadays, even hyperparameter tuning can be programmable by using pipelines. Using relatively basic models, programmers can create and test domain-specific models at an inconceivable speed. Yet apart from the mathematical foundations that ML algorithms rely on, most of the computation happens on the fly and is not stored in a hard-drive. Consequently, humans can not interpret why ML models make their choices. It is just the best answer.

As the topic of my analysis, I used KONDA's data for violence. Hereby in Chapter 1, I will present my design, and in Chapter 2 I will talk about the preprocessing steps and difficulties of my research design.

2: Research Design

2. A. Data Source

In my research, I used a survey made by the KONDA Research and Consultancy Firm conducted in July 2023. KONDA is an established research company that has conducted public opinion polls since 1986. More recently, they have been executing monthly research named "Barometre" since 2008. The survey I am using is themed "childhood", and there are questions regarding childhood experience. Generally speaking, the "Barometre" has a design system that consists of three parts. One part is focused on demographic questions. The other part is focused on political opinion. These two parts are generally unchanged throughout the time, there are only minor changes regarding age-mean or income-mean shifts and approaching political election types. There is a third part that is changing, sometimes by the actual events and sometimes by personal choices. For example, Turkey's recent events are sometimes added as questions in this part. Other times, there are thematic choices such as environmental opinion or lifestyle choices.

The "childhood" theme is incorporated in the last part of the July 2023 research. This part consists of experiences about people's childhood. Such as childhood happiness, childhood popularity, caretakers that people grew up with, discrimination, and violence experience. Since it is very difficult to conduct a survey on children, it is important to see this research. Violence, however, is a confusing social phenomenon. It can be highly detrimental to a child's mental situation. However, there are many Turkish idioms on child education that indicate violence as a method, and thus used as a traditional application of manners. Therefore, I avoid making any claims about the effects of violence in childhood, because I have not any information about the context of the violence. Besides, the research does not present any information on the severity or regularity of violence.

2. B. Preprocessing

KONDA shared their survey data from 2008 to their very recent research. The dataset does not contain a single Barometre research. Therefore we need to get the targeted survey by the Barometre index. In my case, the July 2023 survey was numbered 144. This is fairly the easy part.

What is harder, however, because that the dataset contains many more (as much as an Excel sheet can take) questions from other surveys. On top of that, we have synthetic variables that they derived from the question, such as age or income groupings. Lastly, all of the questions' names are coded by their appearance on the initial questionnaire, and therefore hand made picking is unavoidable. Why dropping empty columns wasn't an option? Because later on, I needed to again analyze the filled columns for their human-readable content. Therefore, I just looked for the columns I was looking for, also eliminating political opinion questions and synthetic variables along the way by this method. After picking my targeted questions, I imported the questionnaire by only these columns and later on I dropped empty columns and rows with empty answers (coded as 99 or blank). Then, I changed column names to human-readable content for analysis. Furthermore, some questions implied the target answers, for example from whom they experienced violence, and these columns were dropped accordingly. These steps were supported by their question name's coding manual. Lastly, I changed column types to integers for applying machine learning algorithms.

Before using any statistical method, I printed distribution graphs to see what the data I was using in my analysis. I also conducted a contingency analysis by the Pearson coefficient, to think about the variables that I am going to use in a traditional part of my analysis. Since I have imbalanced data on childhood experience, I stratified training and split by this.

Later on, when I was testing machine learning algorithms, I also oversampled my training data after I split it. This is a crucial part, oversampling the data before splitting will create data leakage. This would cause a false interpretation of my results. Oversampling was useful in increasing the F1 scores by a couple of percent.

3. Analysis

3. A. Traditional Approach to Machine Learning

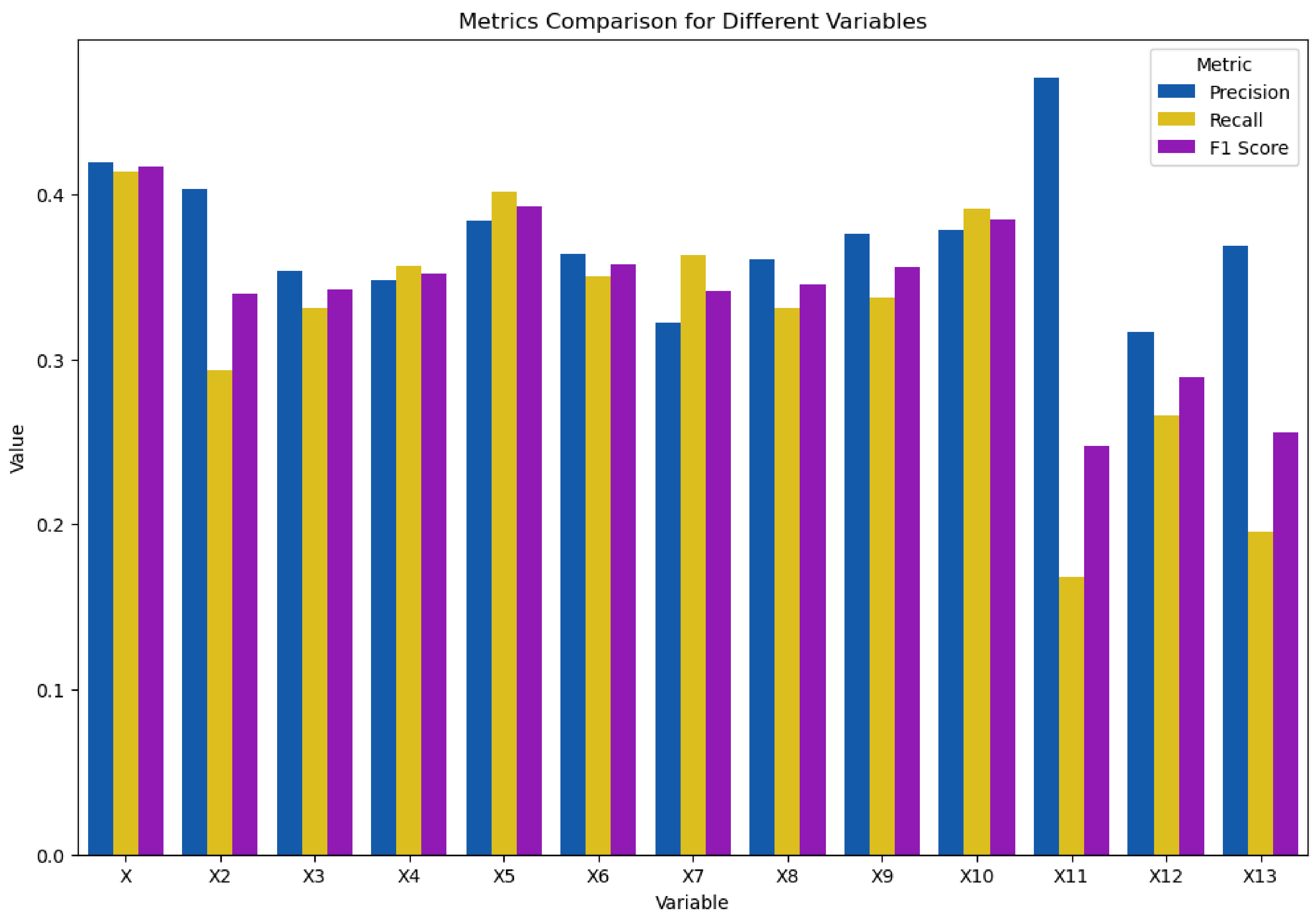
A traditional econometrics approach would focus on the qualitative assumption of explanatory variables. From that departure, it would apply F-tests for explanatory strength of variables and robustness checks for the independence of error term and research design. This part is very much expertise-dependent, because there may be specific situations related to recent events (eg. time) or individual behaviors (eg. omittance). However, in return we have a model that can be refuted or accepted, meaning it is at least critiquable. Basically, the econometrics approach would call for a hypothesis foundation with an extensive explanation of the chosen model.

A machine learning approach on the other hand, when fed with the data, does not necessitate much hypothesis model. The machine learning algorithms simply "converge" to a best-scored equation by the use of training data. The only thing we can do is tuning parameters for the question at hand, and with just enough computation, we can tackle this by programming as well. Sometimes, we can decompose the final results' choices for solving the question but we cannot understand the steps undergone in most of the cases (to understand, the computation power goes n-fold with a vast amount of storage necessity; basically the difference between on-the-fly creation and hard-storage).

For the goal of my research, I created thirteen explanatory variable subsets from my whole set (Named through X1-X13). My base variable set is left mostly unchanged, with only some parts trimmed: for example, numerical age and income are dropped and used .10 quantile groupings, the birth place is dropped and used regions, this was due to tackling extreme values and unrepresentative classes. I used Multi-Layer Perceptron as a machine learning algorithm. Multi-Layer Perceptron (MLP) is one of the first neural network models, dating back to the 1950s.

As my train and test splits, I used a generally accepted 70% to 30% division. However, this part is very much debatable due to the representativity of the test and training set for external applications. Thus applying this by hand, we may consider consulting to theory. However, the general rule would be a bigger test set is always better. In the second part of my research, where I compare machine learning algorithms, I will use 50% splits for training and test sets.

Without further ado, I hereby present the results of my analysis. On the next page, you will see the related variable sets used in the comparison. I visualized the precision, recall, and F1 scores for comparison. As a support to my initial thesis, I was able to reach a nearly good F1 score for a model used in multi-layer perceptron. As a return, I have sets of variables that I can focus on for deeper analysis.


Metrics Comparison for Different Variables
Metric
Precision
Recall
F1 Score
Value
0.0
0.1
0.2
0.3
0.4
X
X2
X3
X4
X5
X6
X7
X8
X9
X10
X11
X12
X13
Variable

Below, I share the set of my variables. We can understand that sex, ethnicity, income group, and birth region might be related to having childhood experience. For a better use of computational methods in combination with the traditional approach, I would suggest increasing variable subsets.

X1 = "Home_popGrouped", "Life_style", "Purdah", "Ethnicity", "Religion", "Age_group", "Income_group"

X2 = "Sex", "Educ", "Growing_place", "Ethnicity", "Happiness", "Childhood_discEthni", "Childhood_discRelig", "Childhood_discLang", "Childhood_discEcon", "Childhood_discGender"

X3 = "Sex", "Ethnicity", "Birth_place", "Growing_place", "Childhood_age", "Age_group"

X4 = "Sex", "Ethnicity", "Birth_place", "Growing_place", "Age_group", "Income_group"

X5 = "Sex", "Ethnicity", "Birth_place", "Growing_place", "Mother_educ", "Father_educ", "Educ"

X6 = "Sex", "Ethnicity", "Income_group", "Educ", "Life_style", "Religiosity", "Childhood_work"

X7 = "Sex", "Ethnicity", "Income_group", "Educ", "Life_style", "Religiosity", "Childhood_age"

X8 = "Sex", "Ethnicity", "Birth_placeRegion", "Home_econ", "Growing_place", "Life_style", "Mother_educ","Childhood_age"

X9 = "Sex", "Ethnicity", "Birth_placeRegion", "Home_econ", "Growing_place", "Life_style", "Mother_educ"

X10 = "Childhood_familyValue", "Childhood_popularity", "Childhood_familyAttitude", "Childhood_familyAttitude", "Childhood_apprec", "Childhood_love"

X11 = "Childhood_relatives", "Childhood_grandpar", "Childhood_sibl", "Childhood_dad", "Childhood_mom", "Childhood_love"

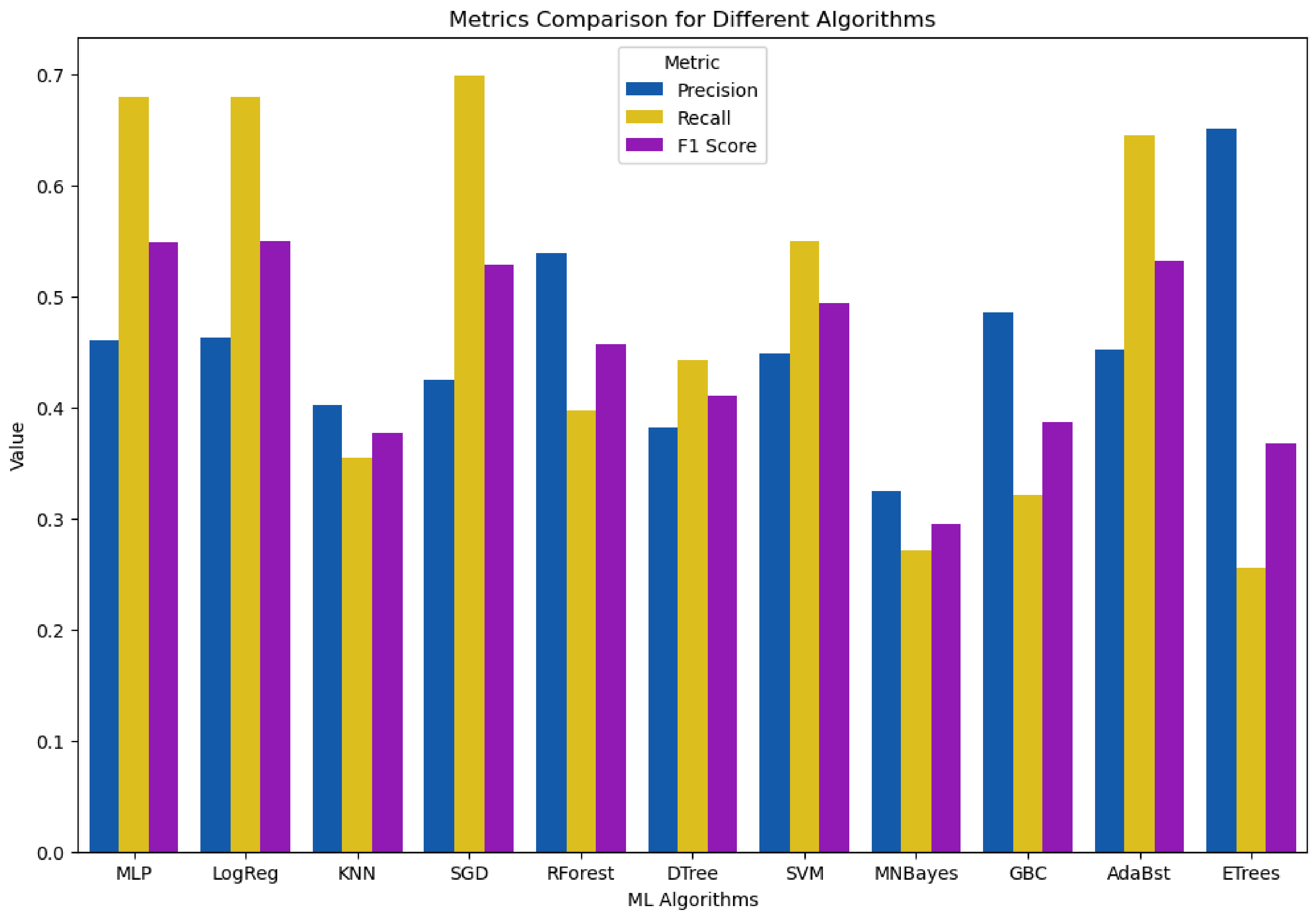
X12 = "Growing_place", "Life_style", "Ethnicity", "Mother_educ", "Father_educ", "Childhood_fam"

X13 = "Sex", "Life_style", "Ethnicity", "Mother_educ", "Father_educ"

3. B. Comparison of Machine Learning Models

In this part, I focused on using machine learning models in a more computational manner. Apart from hyperparameter tuning, which I executed by hand using the SKLearn manual, StackExchange discussions, and data science blog posts, I wanted to see what classification algorithms perform. For this reason, I've used ten more ML algorithms along with multilayer perceptron.

Machine learning use is very easy. We can implement different algorithms with only a few lines of code. For this reason, it is said to be a "spray and pray" kind of approach. We extensively test different algorithms with different hyperparameters and use whatever yields the best score. In my research I used, Logistic Regression, Stochastic Gradient Classifier, Gradient Boosting Classifier, AdaBoost Classifier, Random Forest Classifier, K-Neighbors Classifier, Support Vector Machines, Multinomial Naive Bayes, Decision Tree Classifier, and Extremely Randomized Trees Classifier.


Metrics Comparison for Different Algorithms
Metric
Precision
Recall
F1 Score
Value
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
MLP
LogReg
KNN
SGD
RForest
DTree
SVM
MNBayes
GBC
AdaBst
ETrees
ML Algorithms

4. Hyperparameter Testing and Cross-Validation

This was the last addition to my research. The reason it is applied last is because it is computationally the most intensive method. Here, I created a pipeline to loop through all parameters for all classifier algorithms.

Tests were taking too much time, hereby represent the source code that aimed to be used in the hyperparameter analysis.

```
results_list = []

# Define the scoring metric (F1 score for target value 2)
f1_scorer = make_scorer(f1_score, pos_label=2)

classifiers = {
    'MLPClassifier': Pipeline([
        ('scaler', StandardScaler()),
        ('mlp_classifier', MLPClassifier())
    ]),
    'Logistic Regression': Pipeline([
        ('scaler', StandardScaler()),
        ('logistic_regression', LogisticRegression())
    ]),
    'SGDClassifier': Pipeline([
        ('scaler', StandardScaler()),
        ('sgd_classifier', SGDClassifier())
    ]),
    'GradientBoostingClassifier': Pipeline([
        ('scaler', StandardScaler()),
        ('gradient_boosting', GradientBoostingClassifier())
    ]),
    'AdaBoostClassifier': Pipeline([
        ('scaler', StandardScaler()),
        ('adaboost', AdaBoostClassifier())
    ]),
    'RandomForestClassifier': Pipeline([
        ('scaler', StandardScaler()),
        ('random_forest', RandomForestClassifier())
    ]),
    'KNeighborsClassifier': Pipeline([
        ('scaler', StandardScaler()),
        ('kneighbors', KNeighborsClassifier())
    ]),
    'SVM': Pipeline([
        ('scaler', StandardScaler()),
        ('svm', svm.SVC(kernel='rbf'))
    ]),
    'MultinomialNB': Pipeline([
        ('multinomial_nb', MultinomialNB())
    ]),
    'DecisionTreeClassifier': Pipeline([
        ('scaler', StandardScaler()),
        ('decision_tree', DecisionTreeClassifier())
    ]),
    'ExtraTreesClassifier': Pipeline([
        ('scaler', StandardScaler()),
        ('extra_trees', ExtraTreesClassifier())
    ])
}

# Define the hyperparameter grids for each classifier
hyperparameter_grids = {
    'MLPClassifier': {
        'mlp_classifier__hidden_layer_sizes': [(100,), (100, 100, 100)],
        'mlp_classifier__max_iter': [1000, 2000, 5000],
        'mlp_classifier__activation': ['identity', 'logistic', 'tanh', 'relu'],
        'mlp_classifier__solver': ['lbfgs', 'sgd', 'adam']
    },
    'Logistic Regression': {
        'logistic_regression__C': [0.001, 0.01, 0.1, 1, 10, 100],
        'logistic_regression__penalty': [None, 'l2']
    },
    'SGDClassifier': {
        'sgd_classifier__alpha': [0.1],
        'sgd_classifier__max_iter': [100],
        'sgd_classifier__loss': ['modified_huber']
    },
    'GradientBoostingClassifier': {
        'gradient_boosting__n_estimators': [50],
        'gradient_boosting__learning_rate': [0.2]
    },
    'AdaBoostClassifier': {
        'adaboost__n_estimators': [50],
        'adaboost__learning_rate': [0.2]
    },
    'RandomForestClassifier': {
        'random_forest__n_estimators': [50],
        'random_forest__max_depth': [None]
    },
    'KNeighborsClassifier': {
        'kneighbors__n_neighbors': [3],
        'kneighbors__weights': ['uniform']
    },
    'SVM': {
        'svm__C': [0.1],
        'svm__kernel': ['linear']
    },
    'MultinomialNB': {
        'multinomial_nb__alpha': [0.1],
        'multinomial_nb__fit_prior': [True, False],
        'multinomial_nb__class_prior': [None, [0.7, 0.3]]
    },
    'DecisionTreeClassifier': {
        'decision_tree__max_depth': [None],
        'decision_tree__min_samples_split': [2, 5]
    },
    'ExtraTreesClassifier': {
        'extra_trees__n_estimators': [50],
        'extra_trees__max_depth': [None]
    }
}

# Create a List to store results
results_list = []
```

```
# Loop through classifiers and perform hyperparameter tuning
for name, classifier in classifiers.items():
    # Create GridSearchCV instance for each classifier
    grid_search = GridSearchCV(classifier, hyperparameter_grids[name], cv=5, scoring=f1_scorer,
return_train_score=True)

    # Fit GridSearchCV on the training set (X_train and y_train)
    grid_search.fit(X_train_CV, y_train_CV)

    # Retrieve best hyperparameters
    best_params = grid_search.best_params_

    # Evaluate the best model on the training set
    best_model = grid_search.best_estimator_
    y_train_pred = best_model.predict(X_train_CV)

    # Evaluate the best model on the test set
    y_test_pred = best_model.predict(X_test_CV)

    # Store results in the List
    results_list.append({
        'Classifier': name,
        'Hyperparameters': best_params,
        'Train F1 Score': f1_score(y_train_CV, y_train_pred, pos_label=2),
        'Test F1 Score': f1_score(y_test_CV, y_test_pred, pos_label=2)
    })

    plt.figure(figsize=(8, 6))
    sns.lineplot(x=[str(params) for params in grid_search.cv_results_['params']],
y=grid_search.cv_results_['mean_test_score'], label='Test F1 Score')
    sns.lineplot(x=[str(params) for params in grid_search.cv_results_['params']],
y=grid_search.cv_results_['mean_train_score'], label='Train F1 Score')
    plt.title(f'Hyperparameter Tuning for {name}')
    plt.xlabel('Hyperparameters')
    plt.ylabel('Mean F1 Score (CV)')
    plt.xticks(rotation=45, ha='right')
    plt.legend()
    plt.tight_layout()
    plt.show()


# Create DataFrame from the list of results
results_df = pd.DataFrame(results_list)
# Visualize the results using bar plots
plt.figure(figsize=(15, 8))

sns.barplot(x='Train F1 Score', y='Classifier', data=results_df, color='blue', label='Train F1 Score')
sns.barplot(x='Test F1 Score', y='Classifier', data=results_df, color='orange', label='Test F1 Score')
plt.title('F1 Score for Target Value 2 - Hyperparameter Tuning Results')
plt.xlabel('F1 Score')
plt.ylabel('Classifier')
plt.legend()
plt.tight_layout()
plt.show()

print(results_df)
```

5. Conclusion

In conclusion, I hereby present an approach to analyzing a survey using computationally assisted traditional methods and purely computational methods. In our data, we saw around 30% occurrence of childhood violence experience. In our last assessments, we were able to get around 42% precision with %60 recall, reaching 55% F1 score. For this reason, it is seen that machine learning algorithms help to predict better than random selection. We also used a sample of 1700 people with a 50% train-test split. Therefore, we can conclude that machine learning algorithms work fairly well with a low amount of training data for assessing out-of-sample observations.

In the resume, Chapter 1 provided an explanatory assessment of our data. In Chapter 2, I compared a neural network algorithm with hand-made variable subsets. In Chapter 3, I compared different classification algorithms. In Chapter 4, I provided an example of hyperparameter comparison using extensive computation.