Project 1: Machine Learning with Alzheimer Disease Data

Yoonjoung Kim

2025-02-17

Contents

1 Introduction

During the Fall 2024 Data Mining course (CS 5310) under Professor Pablo Emilio Guillen Rondon, I worked on a project aimed at replicating Experiments 2 and 3 from the paper titled "An Explainable Machine Learning Approach for Alzheimer's Disease Classification" by Abbas Saad Alatrany et al. Additionally, the project aimed to improve classification accuracy by utilizing AutoML. The study utilized the "investigator_nacc67.csv" dataset from the National Alzheimer's Coordinating Center(NACC), which includes 19,519 records and 1,024 features, to classify normal cognition (NC), mild cognitive impairment (MCI), and Alzheimer's disease (AD). Exploratory data analysis (EDA) was done with Ydata Profiling, and utilized Boruta Algorithm for feature selection. The modified data with selected features was classified using AutoML TPOT library.

2 Background

2.1 Automated Machine Learning (AutoML)

Automated Machine Learning(AutoML) is a hot new field. The goal of AutoML is to make machine learning more accessible by automatically generating a data analysis pipeline that can include data pre-processing, feature selection, and feature engineering methods along with machine learning methods and parameter settings that are optimized for your data. Each of these steps can be time-consuming for the machine learning expert and can be debilitating for the novice. By automating this process allows data scientists save time and energy to deal with a large data set. Notable examples include Auto-WEKA, which uses Bayesian optimization; TPOT, known for its genetic algorithms; and Auto-sklearn, which builds on scikit-learn. H2O.ai offers a scalable open-source platform, while DataRobot provides a comprehensive commercial solution. MLJAR focuses on simplicity and accessibility, catering to both beginners and experts. These tools make machine learning more accessible and efficient for a wide range of users.

Below(Figure 1) is an example of a hypothetical machine learning pipeline that could be discovered using a method such as TPOT(AutoML 2025) that I used in this project. Here, the data are analyzed using a random forest (RF) with feature selection performed using the importance scores. The selected features then undergo a polynomial transformation before being analyzed using k nearest neighbors (KNN). The predictions made by KNN are then treated as a new engineered feature and passed to a decision tree (DT). In parallel, the data are also engineered using principal components analysis (PCA). The principal components are then passed as new features to a support vector machine (SVM) whose output is passed as an engineered feature to the DT with the other engineered feature. The DT then makes a final prediction. Each of the methods in this pipeline are included in the scikit-learn library.

2.2 Early Detection of Alzheimer's Disease

Early detection of Alzheimer's disease (AD) remains a challenge, largely due to the nuanced biomarker alterations that frequently go unnoticed(Alatrany et al. 2024). Nonetheless, machine learning (ML) methodologies have emerged as a powerful way for identifying individuals at heightened risk of developing AD. There are no drug that can reverse the progress of AD yet. However, certain drugs can help slow the progression of Alzheimer's disease, especially when administered in the early stages. Thus, It's very critical to detect AD at an early stage.

3 Dataset

In this study, we utilized a comprehensive dataset("investigator_nacc67.csv") acquired from the National Alzheimer's Coordinating Center(NACC), comprising 195,196 records and 1,024 features as of November

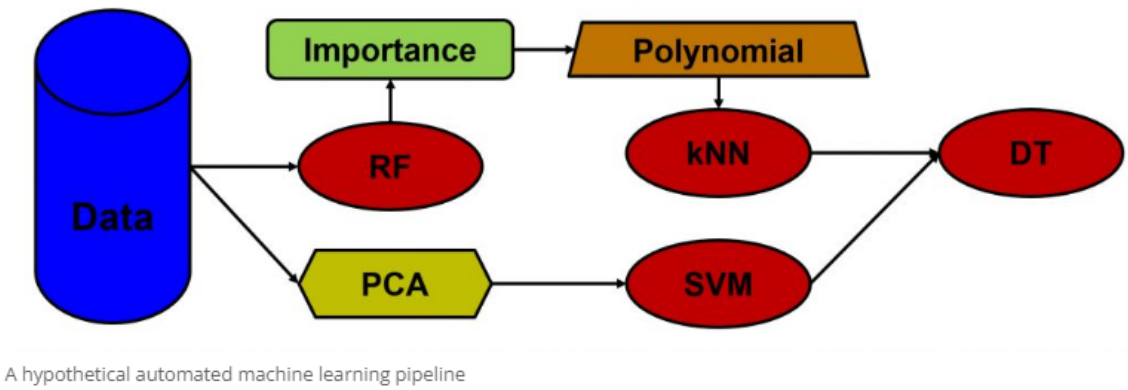

Figure 1: AutoML Learning Pipeline

2024 for classification differentiating between normal cognition(NC), Mild cognitive impairment(MCI), and Alzheimer Disease(AD). From this extensive dataset, we identified critical features and implemented an automated machine learning (AutoML) approach using TPOT AutoML library to optimize the classification process and reproduce the experiments 2 and 3 and then compared with the result in the paper.

```
NACCVNUM variable
NACCVNUM=1 : initial visit\
NACCVNUM=2 :first follow-up completed, etc.
VISITMO, VISITDAY, and VISITYR : visit date
NACCFDYS: days since Initial Visit
```

Data availability: The NACC Dataset is openly available for researchers at https://naccdata.org/ with a data access request.

3.1 EDA

Exploratory data analysis (EDA) was done utilizing Ydata Profiling, formerly known as Pandas Profiling, is a powerful tool for data profiling and exploratory data analysis (EDA). It automates the generation of detailed reports that include a wide range of statistics and visualizations, providing a comprehensive overview of your data set.

3.1.1 Key Features of Ydata Profiling

- **Type Inference**: Automatically detects data types (categorical, numerical, date, etc.).
- **Warnings**: Highlights potential data issues like missing values, skewness, and duplicates.
- **Univariate Analysis**: Provides descriptive statistics and visualizations for individual variables.
- **Multivariate Analysis**: Examines relationships between variables, including correlations and pairwise interactions.
- **Time-Series and Text Analysis**: Offers specialized analysis for time-dependent data and text data.
- **Flexible Output Formats**: Reports can be exported as HTML, JSON, or integrated as widgets in Jupyter Notebooks. (YData Profiling, n.d.)

3.2 Feature importance and selection

3.2.1 Boruta algorithm

The Boruta algorithm is a technique used to identify all relevant features in a dataset by utilizing the Random Forest classification algorithm. It starts by training a Random Forest model to assess the importance of each feature. To create a baseline for comparison, Boruta generates shadow features by randomly shuffling the values of the original features. These shadow features help determine the significance of the original features.

The algorithm then iteratively compares the importance of the original features with the shadow features. Features that are less important than the most significant shadow feature are considered irrelevant and are removed. This process continues until all features are either confirmed as important or deemed irrelevant. Boruta is particularly useful because it aims to identify all relevant features, offering a thorough feature selection method.

Data subset	Selected features
NC vs AD	MEMORY, ORIENT, JUDGMENT, COMMUN, PERSCARE, COMPORT, CDRLANG, MEMPROB, NACCGDS, AGIT, ANX, APA, IRR, MOT, BILLS, TAXES, SHOPPING, GAMES, STOVE, MEALPREP, EVENTS,PAYATTN, TRAVEL, INDEPEND
NC vs MCI	MEMORY, ORIENT, JUDGMENT, COMMUN, HOMEHOBB, CDRLANG, MEMPROB, NACCGDS, BILLS, TAXES
MCI vs AD	MEMORY, ORIENT, JUDGMENT, COMMUN, HOMEHOBB, PERSCARE, CDRLANG, NACCGDS, BILLS, TAXES, SHOPPING, GAMES, MEALPREP, EVENTS, PAYATTN, TRAVEL, INDEPEND
NC vs MCI vs AD	MEMORY, ORIENT, JUDGMENT, COMMUN, HOMEHOBB, CDRLANG, MEMPROB, NACCGDS, BILLS, TAXES, SHOPPING, GAMES, STOVE, MEALPREP, EVENTS, PAYATTN, TRAVEL, INDEPEND

Figure 2: Informative features selected

4 Classification with TPOT AutoML

4.1 What is TPOT AutoML?

TPOT (Tree-based Pipeline Optimization Tool) is an open-source AutoML (Automated Machine Learning) library in Python. It leverages genetic programming to optimize machine learning pipelines, making it easier to discover the best model for a given dataset. TPOT automates the process of selecting and tuning machine learning models, as well as preprocessing steps, by intelligently exploring thousands of possible pipelines.

TPOT is built on top of the popular Scikit-Learn library, which means it integrates seamlessly with many existing machine learning tools and workflows. By automating the most tedious parts of machine learning, TPOT allows data scientists and machine learning practitioners to focus more on understanding their data and less on the trial-and-error process of model selection and hyperparameter tuning.

TPOT will automate the most tedious part of machine learning by intelligently exploring thousands of possible pipelines to find the best one for your data. Once TPOT is finished searching, it provides you with the Python code for the best pipeline it found so you can tinker with the pipeline from there.

4.2 Summary of Original Experiments

The dataset is divided into 80 percent of samples for training while the rest of the samples are reserved to evaluate the model (i.e., unseen data).

Experiment 1 (EXP1) of the paper is essential for determining the efficacy of ML models in classifying clinical stages of cognitive impairment using the NACC dataset. To ensure accurate results, pre-processing steps must be conducted first followed by the training and evaluation of ML models.is experiment is conducted

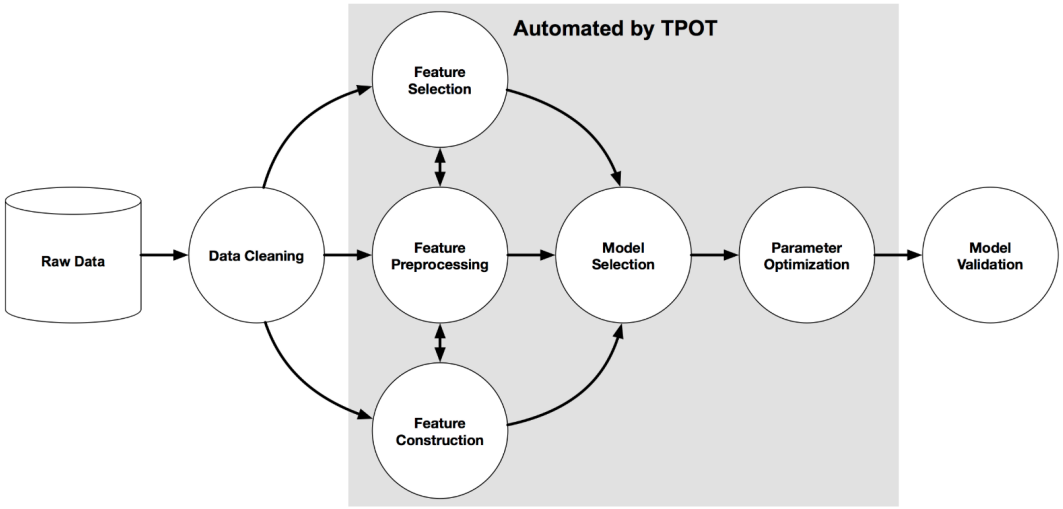

Figure 3: TPOT AutoML

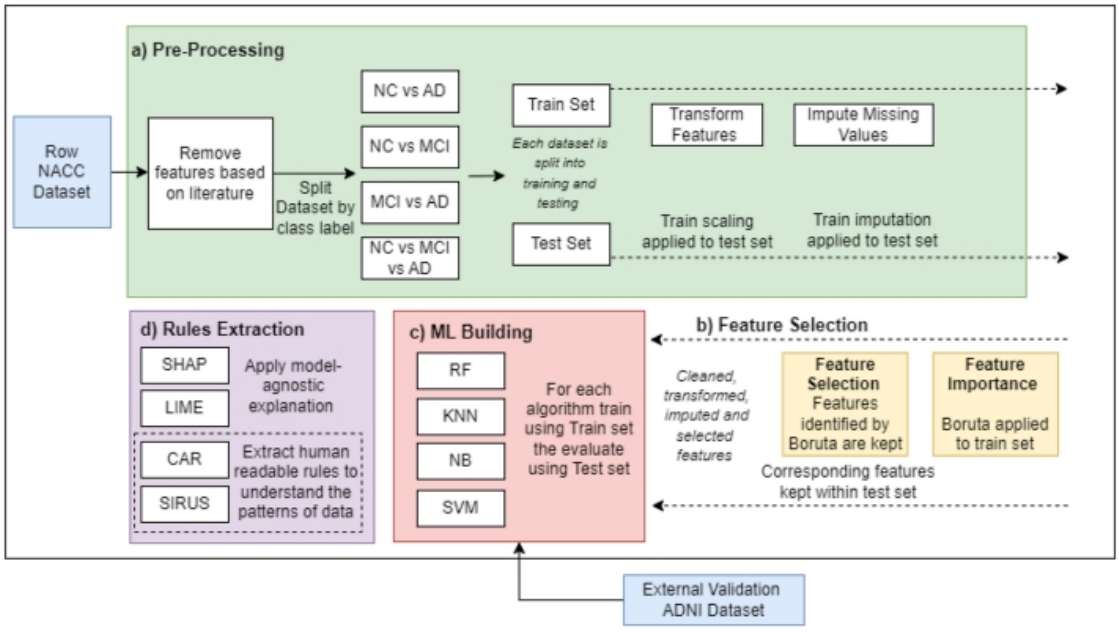

Figure 4: Workflow Overview

to assess the ability of ML models to accurately classify the clinical stages with 64, 55, 67, 66 features (i.e. before feature selection) for NC vs AD, NC vs MCI, MCI vs AD and NC vs MCI vs AD, respectively.

In Experiment 2 (EXP2), ML models with same configuration as EXP1 are trained and evaluated over the reduced feature set (i.e., using the only important features only as identified by Algorithm 1 resulting in selecting 24 feature for NC vs AD subset, 10 for NC vs MCI subset, 17 for MCI vs AD subset and 18 for NC vs MCI vs AD subset."is experiment is performed to determine the effectiveness of our feature selection algorithm for the purpose of classifying AD.

Experiment 3 (EXP3) is a continuation of EXP2, in which the same features are used to train and test the ML models for each data subset. However, instead of classifying the cognitive state of an individual at the baseline visit, the task of the ML models is to predict the cognitive state of an individual after four years of baseline visit. This is a crucial experiment, as it will help to determine whether the features that have been identified as important are effective in predicting the cognitive state of an individual over a longer period of time.

4.3 EXP2 and EXP3 utilizing TPOT AutoML

The verbosity parameter in TPOT AutoML controls the level of detail in the logs, ranging from 0 (no output) to 3+ (very detailed output, including pipeline configurations). The max_time_mins parameter limits the total runtime of the optimization process to 5 minutes, which is useful for quickly testing TPOT's functionality or avoiding long runs during development. This allows users to balance between getting sufficient information and managing the time spent on optimization.

1) Data Preparation:

df1 = pd.DataFrame(CNvsAD): Converts the CNvsAD data into a DataFrame. df1.head(): Displays the first few rows of the DataFrame. df1.shape: Shows the shape of the DataFrame, which has 38,509 rows and 172 columns.

2)Feature and Target Selection:

X1 = df1[CNvsAD_selected_features]: Selects the features for the model. y1 = df1.NACCUDSD: Selects the target variable.

3) Train-Test Split:

X1_train, X1_test, y1_train, y1_test = train_test_split(X1, y1, train_size=0.8, random_state=123): Splits the data into training and testing sets with 80% for training and 20% for testing. X1_train.shape, X1_test.shape, y1_train.shape, y1_test.shape: Displays the shapes of the training and testing sets.

4) TPOT Classifier Setup:

CN_AD = TPOTClassifier(verbosity=2, max_time_mins=5, random_state=123): Initializes the TPOT classifier with a verbosity level of 2 (detailed output) and a maximum runtime of 5 minutes.

5) Model Training:

CN_AD.fit(X1_train, y1_train): Fits the TPOT classifier to the training data.

6) Optimization Process:

The optimization process starts but is interrupted after 5.02 minutes, as specified by max_time_mins=5. A warning is issued that stopping early may not yield the best pipeline. Despite the interruption, TPOT uses the best pipeline found so far. The best pipeline identified includes a GradientBoostingClassifier followed by a DecisionTreeClassifier with specific hyperparameters. This pipeline is used for further predictions and analysis.

4.3.1 EXP2 result

The below is an output of EXP2 of "NC vs AD"

a) NC vs AD - EXP2

```
In [109]: df1 = pd.DataFrame(CNvsAD)

In [111]: df1.head()
          df1.shape

Out[111]: (38509, 172)

In [113]: X1 = df1[CNvsAD_selected_features]
          y1 = df1.NACCUDSD

In [115]: X1_train, X1_test, y1_train, y1_test = train_test_split(X1,y1, train_size =0.8, random_state=123)

In [104]: X1_train.shape, X1_test.shape, y1_train.shape, y1_test.shape

Out[104]: ((30807, 25), (7702, 25), (30807,), (7702,))

In [199]: CN_AD = TPOTClassifier(verbosity=2, max_time_mins=5, random_state=123)
```

verbosity: Sets the level of detail in logs: 0: No output. 1: Minimal output (warnings/errors). 2: More detailed output, including evaluation scores. 3+: Even more detailed output (pipeline configurations, etc.).

max_time_mins=5: Limits the total runtime of the optimization process to 5 minutes. It is useful for quickly testing TPOT functionality or avoiding long runs during development.

```
In [202]: CN_AD.fit(X1_train, y1_train)

Optimization Progress:   0%|          | 0/100 [00:00<?, ?pipeline/s]


5.02 minutes have elapsed. TPOT will close down.
TPOT closed during evaluation in one generation.
WARNING: TPOT may not provide a good pipeline if TPOT is stopped/interrupted in a early generation.


TPOT closed prematurely. Will use the current best pipeline.

Best pipeline: DecisionTreeClassifier(GradientBoostingClassifier(input_matrix, learning_rate=0.1, max_depth=4, max_features=0.9
500000000000001, min_samples_leaf=8, min_samples_split=2, n_estimators=100, subsample=0.9500000000000001), criterion=entropy, m
ax_depth=1, min_samples_leaf=19, min_samples_split=6)

Out[202]: TPOTClassifier(max_time_mins=5, random_state=123, verbosity=2)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
In [204]: print(CN_AD.score(X1_test, y1_test))

0.981303557517528
```

Figure 5: Output of NC vs AD - EXP2

Table 1: Result of EXP2 - TPOT AutoML

EXP2	Best_Pipeline	Accuracy
NC vs AD	Decision Tree	0.981
NC vs MCI	MLPClassifier	0.894
MCI vs AD	Random Forest	0.888
NCvsMCIvsAD	Random Forest	0.862

4.3.2 comparison in Accuracy of Experiment 2

Overall, the accuracy achieved using TPOT AutoML for EXP2 are similar to those reported in the paper, but with higher performance.

Table 2: Accuracy Comparison Original Paper vs TPOT AutoML

EXP2	Original	TPOT_AutoML
NC vs AD	0.975	0.981
NC vs MCI	0.881	0.894
MCI vs AD	0.866	0.888
NCvsMCIvsAD	0.847	0.862

4.3.3 EXP3 result

Table 3: Result of EXP3 - TPOT AutoML

EXP3	Best_Pipeline	Accuracy
NC vs AD	Extra Trees Classifier	0.973
NC vs MCI	Decision Tree	0.773
MCI vs AD	Gradient Boosting Classifier	0.845
NCvsMCIvsAD	Extra Trees Classifier	0.774

4.3.4 Comparison in Accuracy of Experiment 3

The accuracies achieved using TPOT AutoML for EXP3 are comparable to or exceed those reported in the original study. We observed higher performance with TPOT AutoML compared to Scikit-Learn Machine Learning used in the initial research. Specifically, the accuracy for distinguishing MCI from AD improved from 0.782 to 0.845. This significant improvement in accuracy is crucial, as it enables physicians to identify patients with MCI who are likely to progress to AD, allowing for earlier intervention and treatment.

Table 4: Accuracy Comparison Original Paper vs TPOT AutoML

EXP3	Original	TPOT_AutoML
NC vs AD	0.964	0.973
NC vs MCI	0.781	0.773
MCI vs AD	0.782	0.845
NCvsMCIvsAD	0.730	0.774

5 Conclusion

Running TPOT AutoML is quite simple with only a couple of lines of coding, but very effective and powerful. This study demonstrated that TPOT AutoML achieved comparable or superior accuracy to the original research, highlighting its effectiveness in handling large datasets and enhancing classification performance by leveraging machine learning (ML) methodologies, the research addressed the challenges of early Alzheimer's disease (AD) detection. Additionally, the project successfully replicated and expanded upon the original experiments, reaffirming TPOT AutoML's capability to identify key patterns efficiently and generate robust classification models. The comparison with the original study validated TPOT's effectiveness and highlighted the potential of explainable ML tools in advancing early detection and understanding of Alzheimer's disease.

References

Alatrany, Abbas Saad, Wasiq Khan, Abir Hussain, Hoshang Kolivand, and Dhiya Al-Jumeily. 2024. "An Explainable Machine Learning Approach for Alzheimer's Disease Classification." *Scientific Reports* 14 (1): 2637. https://doi.org/10.1038/s41598-024-51985-w.

AutoML. 2025. "AutoML: Automated Machine Learning." https://automl.info/.

YData Profiling. n.d. "YData Profiling." https://ydata-profiling.ydata.ai/.