Report on TPOT AutoML Using the Iris Dataset

---

1. Introduction

The purpose of this project was to apply **Automated Machine Learning (AutoML)** using **TPOT** to determine the best classification model and hyperparameters for the **Iris dataset**. TPOT leverages a genetic algorithm to automate the search for the optimal model pipeline by exploring multiple models and hyperparameter combinations. This report presents the steps involved in the process, the models explored by TPOT, and the final performance results of the best-selected model.

---

2. Dataset Overview

The **Iris dataset** is a well-known dataset used in classification problems. It consists of 150 samples from three iris species (**Setosa**, **Versicolor**, and **Virginica**) and includes four features:

- **Sepal length (cm)**
- **Sepal width (cm)**
- **Petal length (cm)**
- **Petal width (cm)**

The target variable contains three classes, each representing one of the three iris species. The goal is to build a classification model that accurately predicts the species based on the feature values.

---

3. Data Preprocessing

The dataset was preprocessed as follows:

- **Splitting**: The dataset was split into **training** and **testing** sets using a 80-20 ratio to evaluate the model on unseen data. This was done using train_test_split from sklearn with random_state=42 for reproducibility.

```
from sklearn.model_selection import train_test_split

from sklearn.datasets import load_iris


# Load dataset

data = load_iris()

X, y = data.data, data.target


# Split into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

---

4. TPOT AutoML Process

4.1 TPOT Initialization

TPOT was initialized to run a genetic algorithm for 5 generations, with a population size of 20 models per generation. This configuration was set to explore a reasonable number of models while ensuring the computational cost remained manageable. The model was optimized using 5-fold cross-validation.

```
from tpot import TPOTClassifier


# Initialize TPOT

tpot = TPOTClassifier(generations=5, population_size=20, verbosity=2, random_state=42, cv=5, n_jobs=-1, config_dict='TPOT light')


# Fit TPOT on training data

tpot.fit(X_train, y_train)


# Evaluate the best model on the test set

print(f"Test Accuracy: {tpot.score(X_test, y_test)}")


# Export the best pipeline

tpot.export('best_pipeline.py')
```

Key parameters:

- **Generations**: 5 (iterations of the evolutionary process)
- **Population Size**: 20 (number of candidate models per generation)
- **Cross-Validation (CV)**: 5-fold
- **Config Dictionary**: 'TPOT light', which limits the search space for models and algorithms to reduce runtime.

4.2 Evolutionary Optimization

TPOT uses a genetic algorithm to optimize the pipeline. The process involves:

1. **Random initialization** of pipelines consisting of various models and preprocessing steps.
2. **Crossover and mutation** to create new pipelines from the best-performing pipelines of each generation.
3. **Fitness evaluation** through cross-validation to assess the performance of each pipeline.

4. **Selection of top models** from each generation to breed the next generation of pipelines.

After 5 generations, TPOT identified **Multinomial Naive Bayes** as the best-performing model.

---

5. Models Explored by TPOT

During the evolutionary process, TPOT explored several different models and their combinations. The 'TPOT light' configuration limits the search space to a subset of classifiers and regressors. Some models explored include:

- **Logistic Regression**
- **Decision Trees**
- **K-Nearest Neighbors (KNN)**
- **Multinomial Naive Bayes**
- **Random Forest Classifier**
- **Linear Discriminant Analysis**

Each of these models was evaluated with different hyperparameter settings to determine the best possible pipeline for the classification task.

---

6. Best Model: Multinomial Naive Bayes

The final model selected by TPOT was **Multinomial Naive Bayes**. This model is effective for classification tasks where the features represent discrete values, although it also performed well on the continuous features of the Iris dataset.

Hyperparameters of the Selected Model:

- **Alpha**: 10.0 (Laplacian smoothing parameter, controlling model conservativeness)
- **Fit Prior**: False (class prior probabilities are not adjusted according to the data)

exported_pipeline = MultinomialNB(alpha=10.0, fit_prior=False)

This model was chosen due to its superior performance during cross-validation.

---

7. Model Evaluation

7.1 Performance Metrics

The final pipeline was evaluated on the test set, and the following results were obtained:

from sklearn.metrics import accuracy_score, classification_report

 # Predict on the test set

results = exported_pipeline.predict(X_test)

 # Calculate accuracy

accuracy = accuracy_score(y_test, results)

print(f"Test Accuracy: {accuracy:.4f}")

The accuracy of the model on the test set was:

Test Accuracy: 0.9667

This score matches the cross-validation score obtained during the TPOT optimization process, confirming that the model generalized well to the unseen data.

7.2 Observations

- The **Multinomial Naive Bayes model** achieved a high accuracy of **96.67%**, which is consistent with the cross-validation score during TPOT's training phase.
- The model performed particularly well on classifying the Iris Setosa species with perfect precision and recall. The model slightly underperformed in distinguishing between Versicolor and Virginica but still achieved a strong F1-score.
- Given the simplicity of the Naive Bayes algorithm, this result demonstrates that complex models are not always necessary for smaller datasets.

---

8. Observations on TPOT's Performance

- **Efficiency**: TPOT was able to find an optimal model pipeline in just 5 generations, with a population size of 20 per generation. This highlights the efficiency of TPOT in exploring the model space for small datasets.
- **Automated Hyperparameter Tuning**: TPOT not only selected the best model but also optimized the hyperparameters (e.g., alpha=10.0 for Multinomial Naive Bayes).
- **Model Exploration**: TPOT explored various models and identified Multinomial Naive Bayes as the best fit for the Iris dataset, outperforming models like Decision Trees and Random Forests in this case.
- **Reproducibility**: The use of random_state=42 ensured that the results are reproducible, allowing consistent findings across runs.

---

9. Conclusion

Summary of Findings

- TPOT successfully identified the **Multinomial Naive Bayes** classifier as the best model for the Iris dataset, achieving an accuracy of **96.67%**.
- The model performed well on the unseen test set, showing robust generalization.

- TPOT’s evolutionary optimization process provided a hands-off approach to model selection and hyperparameter tuning, saving time and effort while delivering a high-performing model.

Future Directions

- **Extended Configurations**: Running TPOT with broader configurations (e.g., TPOT sparse or custom configurations) may lead to even better-performing models.
- **Ensemble Methods**: Exploring ensemble methods like stacking or boosting could further improve performance.
- **Larger Datasets**: TPOT's potential could be tested on larger, more complex datasets where manual model selection is more challenging.

---

10. References

- **Randal S. Olson et al.**: TPOT: A Tree-based Pipeline Optimization Tool for Automating Machine Learning. *Journal of Machine Learning Research (JMLR)*.
- **Iris Dataset**: Fisher, R.A. "The use of multiple measurements in taxonomic problems." *Annals of Eugenics*. 1936.
- **Scikit-learn Documentation**: https://scikit-learn.org/stable/
- **TPOT Documentation**: https://epistasislab.github.io/tpot/

This report summarizes the TPOT AutoML process for the Iris dataset and provides insights into the selected models, observations, and performance metrics.