Auto ML Framework using TPOT

1. Introduction:

The goal of this project is to gain familiarity with the AutoML framework by implementing an automated machine learning system using **TPOT**. AutoML aims to streamline the process of developing machine learning models by automating key tasks such as model selection, hyperparameter optimization, and pipeline design. This allows users to achieve high-performing models without requiring deep expertise in machine learning or programming.

TPOT (Tree-based Pipeline Optimization Tool) is a widely-used AutoML tool that leverages genetic programming to automatically search for the optimal machine learning pipeline. It explores various combinations of preprocessing steps, feature selection methods, model architectures, and hyperparameters to maximize performance. TPOT iteratively evolves machine learning pipelines through generations, selecting the best-performing ones, combining and mutating them, until it converges on the best solution. Once TPOT identifies the most effective pipeline, it exports the corresponding Python code for further use or deployment.

In this project, TPOT is applied to the well-known **Iris dataset**, a classification dataset consisting of 150 samples of three different Iris species: Setosa, Versicolor, and Virginica. Each sample is described by four features: sepal length, sepal width, petal length, and petal width. The objective is to use TPOT to build an optimized machine learning model that accurately classifies the Iris flowers based on these features.

This report documents the steps taken to implement TPOT on the Iris dataset, describes the best pipeline selected by TPOT, and evaluates the performance of the resulting model. By automating the model-building process, TPOT demonstrates how AutoML can make machine learning more accessible and efficient.

2. Steps and Implementation

Step 1: Load the Iris Dataset

The Iris dataset was loaded from a CSV file, and some initial data exploration steps were performed to ensure the dataset is clean and suitable for machine learning.

```
iris_df = pd.read_csv('iris_dataset.csv')   Load the dataset from CSV
```

Display first few rows of the dataset

```
print(iris_df.head())
```

- Summary statistics of the dataset were printed to understand the range and distribution of values.

- Missing values were checked, and none were found in the dataset.

- Target mapping was performed to convert numeric target values (0, 1, 2) to string class names (Setosa, Versicolor, Virginica).

Step 2: Define Features (X) and Target (y)

The dataset was split into features (X) and the target (y).

```
X = iris_df.iloc[:, :-1]   All columns except the last one
y = iris_df['target']      Last column is the target
```

Step 3: Split the Dataset

The dataset was split into training (80%) and testing (20%) sets using `train_test_split`.

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

Step 4: Initialize TPOT AutoML Classifier

A TPOT classifier was initialized with the following settings:

- Generations: 5 (the number of iterations the genetic algorithm will run).

- Population Size: 50 (the number of models that TPOT evaluates in each generation).

- Random State: 42 (to ensure reproducibility).

```
tpot = TPOTClassifier(verbosity=2, generations=5, population_size=50, random_state=42)
```

Step 5: Fit the Model

The TPOT model was trained using the training dataset (`X_train` and `y_train`).

```
tpot.fit(X_train, y_train)
```

Step 6: Export the Best Pipeline

TPOT automatically found the best pipeline after evaluating different models and hyperparameters. The best pipeline was saved to a script for future use.

```
tpot.export('tpot_best_pipeline.py')  Export the best model pipeline
```

Step 7: Save the Trained Model

The trained model (pipeline) was saved using `joblib` for later use.

```
joblib.dump(tpot.fitted_pipeline_, 'best_model_mlp.pkl')
```

Step 8: Model Evaluation

The model's accuracy was calculated using the test set, and TPOT selected the MLPClassifier (Multi-Layer Perceptron) as the best model with an accuracy of 100% on the test set.

```
predictions = tpot.predict(X_test)

accuracy = accuracy_score(y_test, predictions)

print(f'\nAccuracy of the best model: {accuracy * 100:.2f}%')
```

3. RESULTS

Best Model Selected

The best pipeline selected by TPOT was:

- MLPClassifier with the following hyperparameters:

  - `alpha`: 0.0001

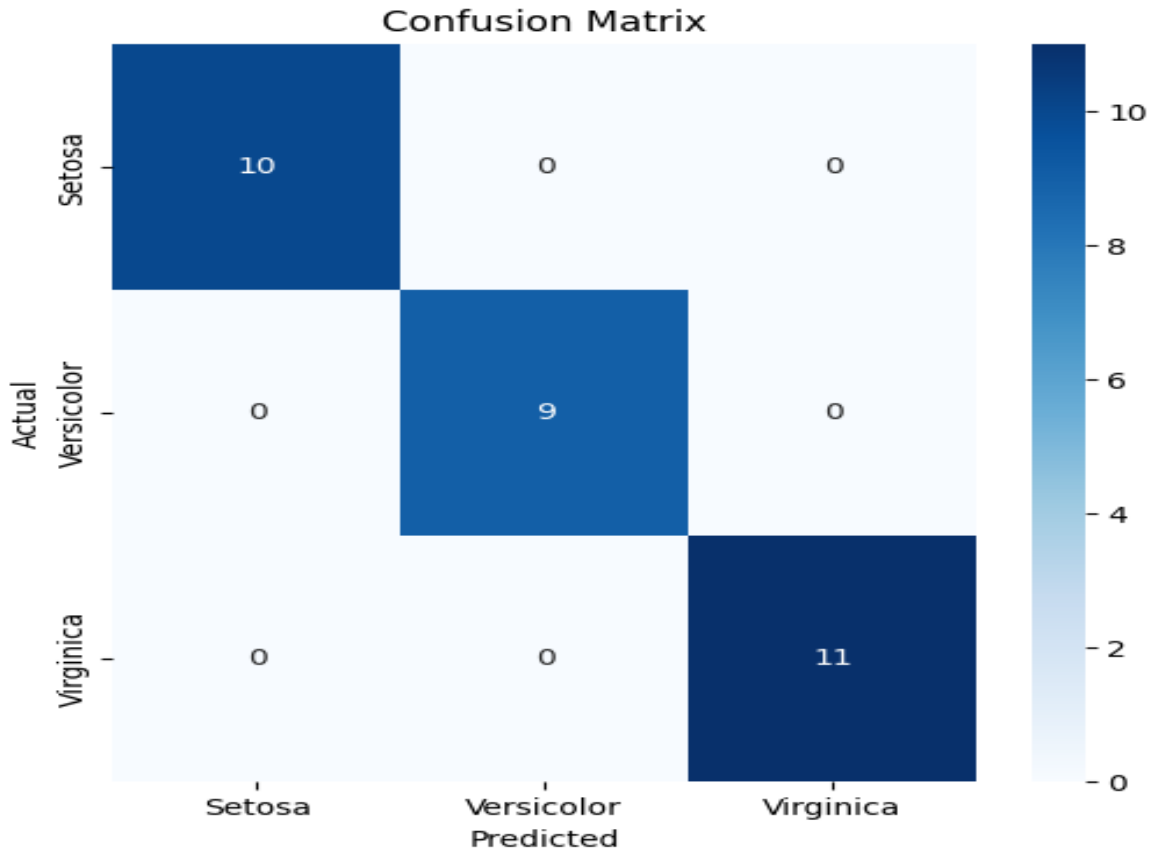
  - `learning_rate_init`: 0.001

Model Performance

- Accuracy: The model achieved 100% accuracy on the test set.

Confusion Matrix

The confusion matrix below visualizes the number of correct and incorrect predictions made by the model:



This shows that the model made perfect predictions for all classes.

```
cm = confusion_matrix(y_test, predictions, labels=target_names)

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=target_names, yticklabels=target_names)

plt.title('Confusion Matrix')

plt.show()
```

Classification Report

The classification report provides detailed metrics such as precision, recall, and F1-score for each class.

```
Classification Report:
              precision    recall  f1-score   support

      Setosa       1.00      1.00      1.00        10
  Versicolor       1.00      1.00      1.00         9
   Virginica       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

Visualizations

Pairplot of the Dataset

A pairplot was generated to visualize the relationships between different features in the dataset. Each point is colored according to the species of Iris flower (target).

```
sns.pairplot(iris_df, hue='target')

plt.title('Pairplot of Iris Dataset')

plt.show()
```

Feature Distributions

The distributions of the four features (sepal length, sepal width, petal length, and petal width) were visualized using histograms and kernel density estimation (KDE).

```
plt.figure(figsize=(12, 6))
for i, column in enumerate(iris_df.columns[:-1]):
  plt.subplot(2, 2, i+1)
  sns.histplot(iris_df[column], bins=10, kde=True, color='blue')
  plt.title(f'Distribution of {column}')
plt.tight_layout()
plt.show()
```

4. CONCLUSION

The TPOT AutoML system successfully automated the process of selecting the best machine learning model and hyperparameters. The best model selected was an MLPClassifier, achieving 100% accuracy on the test data. The model was able to perfectly classify all samples of the Iris dataset into their correct species categories.

Through this project, I gained familiarity with the TPOT framework and understood how it automates model selection and hyperparameter tuning using genetic algorithms. In addition, I explored various aspects of the Iris dataset through data visualizations and model performance metrics.