ASSIGNMENT -1
AUTOML FRAMEWORKS

INTRODUCTION TO AUTOML FRAMEWORKS:

AutoML, or Automated Machine Learning, refers to the process of automating the end-to-end process of applying machine learning to real-world problems. An AutoML framework is a software tool or platform designed to automate various tasks involved in machine learning model development, such as data preprocessing, feature engineering, model selection, hyperparameter tuning, and model deployment.

The main goal of AutoML frameworks is to make machine learning more accessible to users with limited machine learning expertise by automating the complex and time-consuming aspects of the process. This allows users to focus more on the problem they want to solve and less on the intricacies of machine learning algorithms and techniques.

LIST OF SELECTED AUTOML FRAMEWORK:

1. **H2o autoML** - H2O AutoML is an automated machine learning framework that streamlines the model development process. It automates tasks like feature engineering, algorithm selection, and hyperparameter tuning, allowing users to build highly accurate predictive models with minimal manual intervention. H2O AutoML is renowned for its scalability and ease of use.

2. **TPOT -** TPOT is an automated machine learning framework that uses genetic programming to search for the best machine learning pipelines. It explores a wide range of preprocessing techniques and model configurations to find the most effective pipeline for a given dataset, aiming to streamline the model development process.

ABOUT DATASET:

The dataset that is used here is **LAPTOP PRICE PREDICTION.** This dataset provides a comprehensive collection of laptop attributes and features, making it an ideal tool for various analytical and modeling tasks.

Containing information on thousands of laptops, the dataset encompasses a wide range of brands, models, and configurations. It includes both entry-level and high-end laptops, catering to diverse user needs and preferences. Each laptop entry within the dataset offers a plethora of attributes, such as processor details, memory capacity, storage size, display characteristics, graphics capabilities, battery life, operating system, and more.

INPUT FEATURES:

1. **Company:** The manufacturer of the laptop.

2. **TypeName:** The type or category of the laptop, such as Ultrabook, Gaming, Business, etc.

3. **RAM:** Random Access Memory, measured in gigabytes (GB), indicates the memory capacity for running programs.

4. **Weight:** The weight of the laptop, typically measured in kilograms (kg) or pounds (lbs).

5. **TouchScreen:** Binary indicator of whether the laptop has a touchscreen display (1 for yes, 0 for no).

6. **IPS:** Binary indicator of whether the laptop display has In-Plane Switching technology for better color and viewing angles.

7. **PPI:** Pixels Per Inch, a measure of display sharpness or pixel density.

8. **CPU_Brand:** The brand of the Central Processing Unit (CPU) or processor.

9. **HDD:** Hard Disk Drive storage capacity, measured in gigabytes (GB) or terabytes (TB).

10. **SSD:** Solid State Drive storage capacity, typically faster and more reliable than HDD.

11. **GPU_Brand:** The brand of the Graphics Processing Unit (GPU) or graphics card.

12. **OS:** Operating System installed on the laptop, such as Windows, macOS, or Linux.

TARGET FEATURE:

1.Price: The price of the laptop, usually in a specific currency such as dollars ($) or euros (€).

H2o AutoML:

H2O AutoML leverages cutting-edge algorithms and techniques to handle diverse datasets and modeling tasks effectively. It streamlines the entire machine learning pipeline, from data preprocessing to model evaluation, enabling users to focus on problem-solving rather than algorithm selection or parameter tuning. With its scalable and parallelized architecture, H2O AutoML can efficiently handle large datasets and complex modeling scenarios, making it suitable for both beginners and experienced data scientists alike. Additionally, it provides comprehensive model interpretability and visualization tools, empowering users to understand and trust the models generated by the platform.

FEATURES OF H2o AutoML:

1. Automated model selection: H2O AutoML automatically selects and trains the most appropriate machine learning algorithms for a given dataset, reducing the need for manual algorithm selection.

2. Hyperparameter optimization: The platform tunes hyperparameters for each algorithm to maximize model performance, saving time and effort for users.

3. Ensemble learning: H2O AutoML employs ensemble learning techniques to combine predictions from multiple models, often resulting in improved accuracy and robustness.

4. Scalability: It can handle large datasets and complex modeling tasks efficiently, thanks to its distributed computing capabilities and parallelized algorithms.

5. Model interpretability: H2O AutoML provides tools for interpreting model predictions and understanding feature importance, helping users gain insights into their data and models.

6. Time-saving: By automating the end-to-end model building process, H2O AutoML saves users significant time and effort compared to manual model development.

7. Ease of use: The platform is designed to be user-friendly, with simple APIs and interfaces that make it accessible to users with varying levels of machine learning expertise.

8. Extensibility: H2O AutoML can be easily integrated into existing machine learning workflows and pipelines, allowing users to leverage its capabilities alongside other tools and frameworks.

9. Comprehensive reporting: It generates detailed reports and metrics for each trained model, facilitating model evaluation and comparison.

TPOT:

TPOT (Tree-based Pipeline Optimization Tool) is a Python library for automated machine learning. Using genetic programming, it searches for optimal machine learning pipelines, comprising data preprocessing, feature selection, and algorithms. This automation streamlines model development, making it suitable for users with varying levels of expertise. TPOT's scalability allows it to handle datasets of different sizes efficiently. Additionally, its customizable search space and model interpretability features contribute to its versatility and usability in diverse machine learning projects.

FEATURES OF TPOT:

1. Automated pipeline construction: TPOT automates the process of building machine learning pipelines, including feature selection, data preprocessing, and algorithm selection.

2. Genetic programming: TPOT utilizes genetic algorithms to evolve and optimize machine learning pipelines over multiple generations, selecting the best-performing ones.

3. Customizable search space: Users can define constraints and specify the types of preprocessing steps, feature selectors, and algorithms to consider during pipeline optimization.

4. Scalability: TPOT is designed to handle datasets of varying sizes and complexities, making it suitable for small to large-scale machine learning tasks.

5. Model interpretability: TPOT provides insights into the inner workings of generated pipelines, helping users understand feature importance and model behavior.

6. Time-saving: By automating pipeline construction and hyperparameter tuning, TPOT saves significant time and effort in model development.

7. Parallelization: TPOT can leverage multi-core processors to accelerate pipeline optimization, enabling faster experimentation and iteration.

8. Cross-validation: TPOT employs cross-validation techniques to estimate pipeline performance accurately, aiding in avoiding overfitting and ensuring model robustness.

9. Open-source: TPOT is freely available as an open-source project, allowing users to contribute and customize it according to their needs.

10. Integration: TPOT can be seamlessly integrated into existing machine learning workflows and pipelines, facilitating collaboration and leveraging other Python libraries and frameworks.

H2o VS TPOT :

H2O AutoML and TPOT are both automated machine learning frameworks, but they differ in their approaches and features. H2O AutoML focuses on automating the end-to-end process of model selection, hyperparameter optimization, and ensemble learning using a diverse set of machine learning algorithms. It is particularly suitable for users looking for a streamlined and scalable solution with built-in support for distributed computing. In contrast, TPOT employs genetic programming to evolve optimal machine learning pipelines, including feature selection, preprocessing, and algorithm selection. It offers flexibility and customization options, making it ideal for users seeking a more exploratory and adaptable approach to automated machine learning. H2O AutoML's unique features include its comprehensive ensemble learning capabilities, which allow it to combine predictions from multiple models to improve accuracy. Additionally, H2O AutoML offers built-in support for distributed computing, enabling it to handle large-scale datasets and complex modeling tasks efficiently. On the other hand, TPOT's unique features stem from its use of genetic programming to evolve machine learning pipelines. TPOT provides a customizable search space, allowing users to define constraints and specify the types of preprocessing steps, feature selectors, and algorithms to consider during optimization. This flexibility makes TPOT suitable for users who require more control over the automated model building process and want to explore a wide range of possibilities.

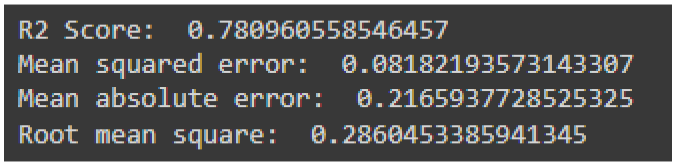
DATA PREPROCESSING:

CODE:

```
import pandas as pd
import numpy as np
import matplotlib as plt
data=pd.read_csv('/content/drive/MyDrive/Laptop - Sheet1.csv')
data.head()
data.tail()
data.isnull().sum()
data.nunique()
data=pd.get_dummies(data)
data
X=data.drop('Price',axis=1)
X
y=data['Price']
y
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,random_state=32,test_size=0.2)
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(X_train,y_train)
```

```
y_pred=lr.predict(X_test)
from sklearn.metrics import
mean_squared_error,mean_absolute_error,r2_score,roc_auc_score,mean_squared_log_error
from sklearn.model_selection import cross_val_predict
print("R2 Score: ",r2_score(y_test,y_pred))
print("Mean squared error: ",mean_squared_error(y_test,y_pred))
print("Mean absolute error: ",mean_absolute_error(y_test,y_pred))
rms = np.sqrt(mean_squared_error(y_test,y_pred))
print("Root mean square: ",rms)
```

<u>OUTPUT FOR MANUAL :</u>



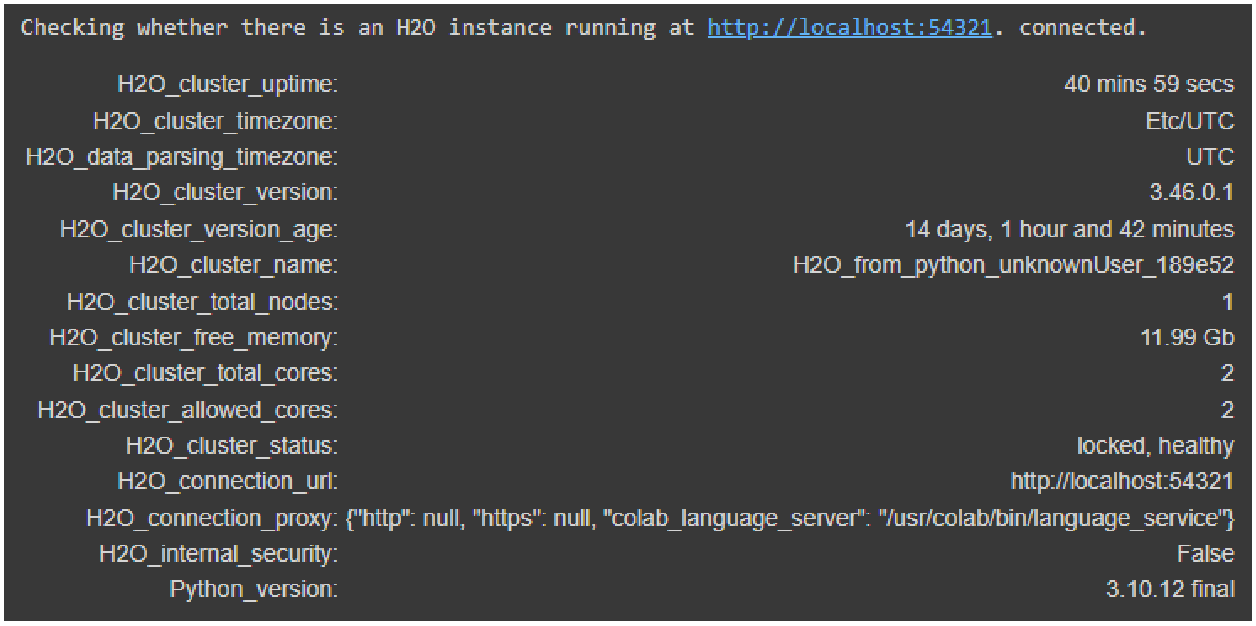
```
R2 Score:  0.780960558546457
Mean squared error:  0.08182193573143307
Mean absolute error:  0.2165937728525325
Root mean square:  0.2860453385941345
```

<u>H2o AutoML:</u>

<u>CODE:</u>

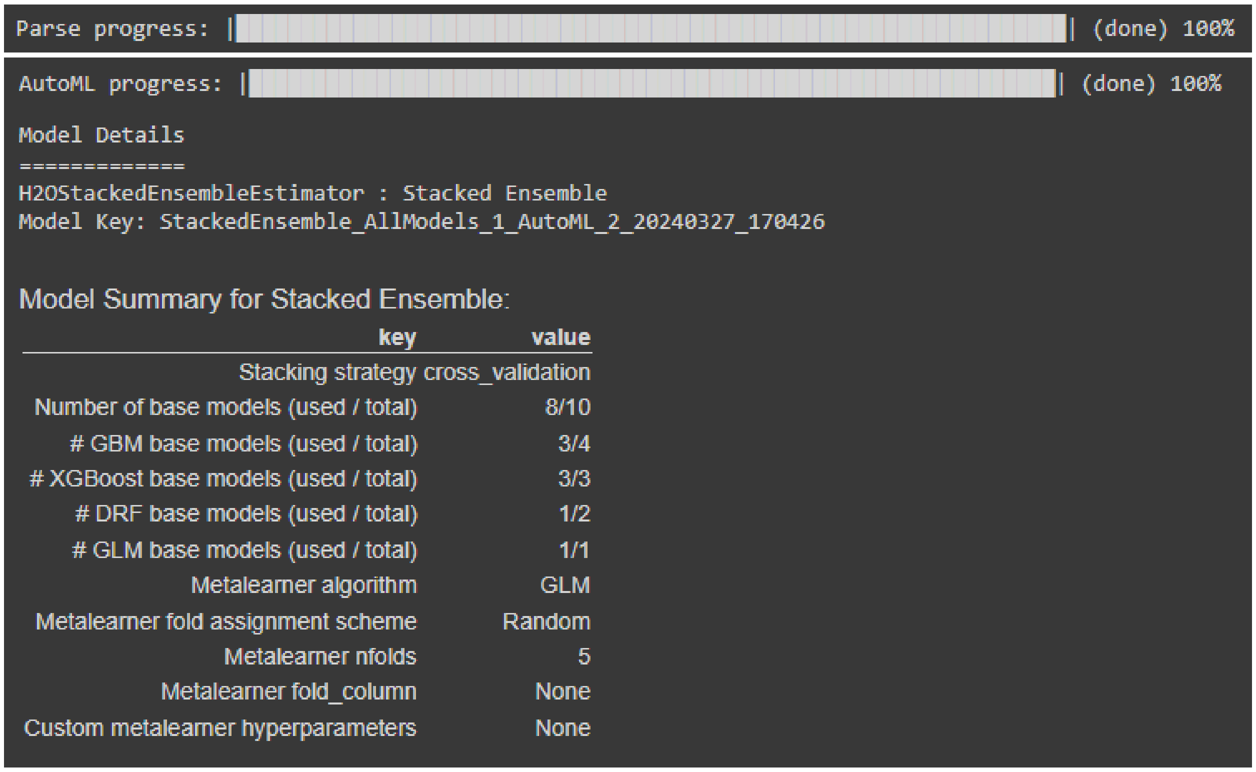
```
!pip install requests
!pip install tabulate
!pip install future
!pip install h2o
import h2o
from h2o.automl import H2OAutoML
h2o.init(nthreads=-1,max_mem_size=12)
train=h2o.H2OFrame(data)
X_intercept=list(X.columns)
y_intercept="Price"
auto_ml = H2OAutoML(max_models=10, seed=1)
auto_ml.train(x=X_intercept, y=y_intercept, training_frame=train)
```

OUTPUT FOR H2o AutoML:



```
Checking whether there is an H2O instance running at http://localhost:54321. connected.
```

H2O_cluster_uptime:	40 mins 59 secs
H2O_cluster_timezone:	Etc/UTC
H2O_data_parsing_timezone:	UTC
H2O_cluster_version:	3.46.0.1
H2O_cluster_version_age:	14 days, 1 hour and 42 minutes
H2O_cluster_name:	H2O_from_python_unknownUser_189e52
H2O_cluster_total_nodes:	1
H2O_cluster_free_memory:	11.99 Gb
H2O_cluster_total_cores:	2
H2O_cluster_allowed_cores:	2
H2O_cluster_status:	locked, healthy
H2O_connection_url:	http://localhost:54321
H2O_connection_proxy:	{"http": null, "https": null, "colab_language_server": "/usr/colab/bin/language_service"}
H2O_internal_security:	False
Python_version:	3.10.12 final

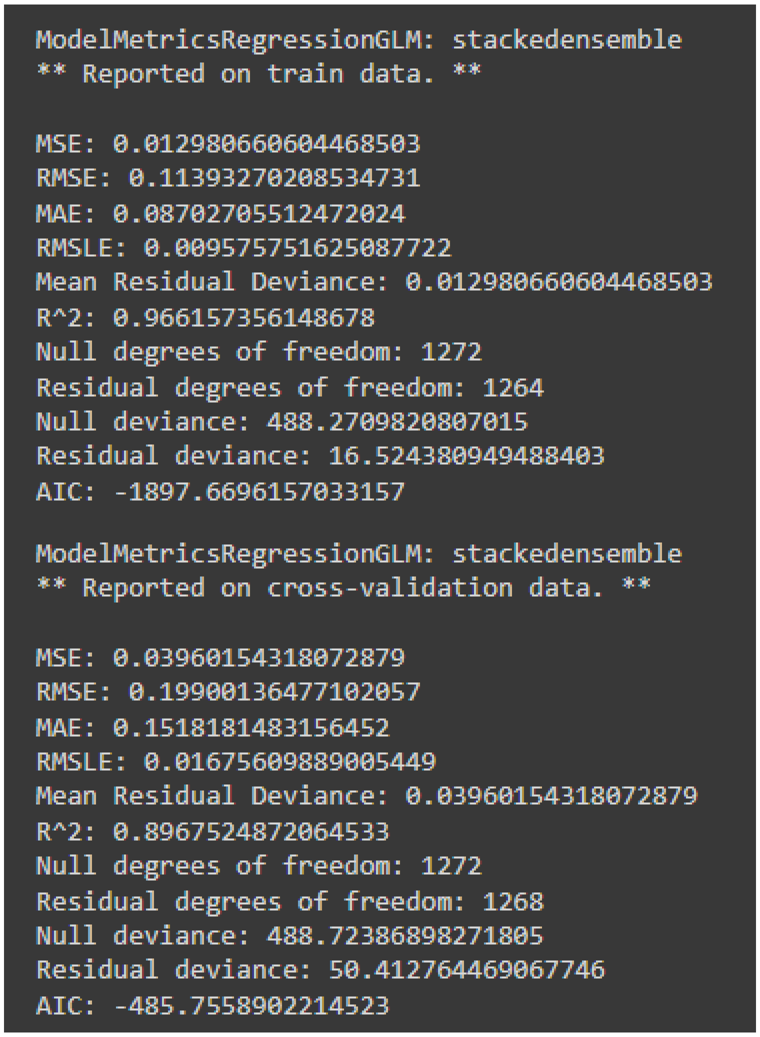


```
Parse progress: |████████████████████████████████████████████████████████████████| (done) 100%
```

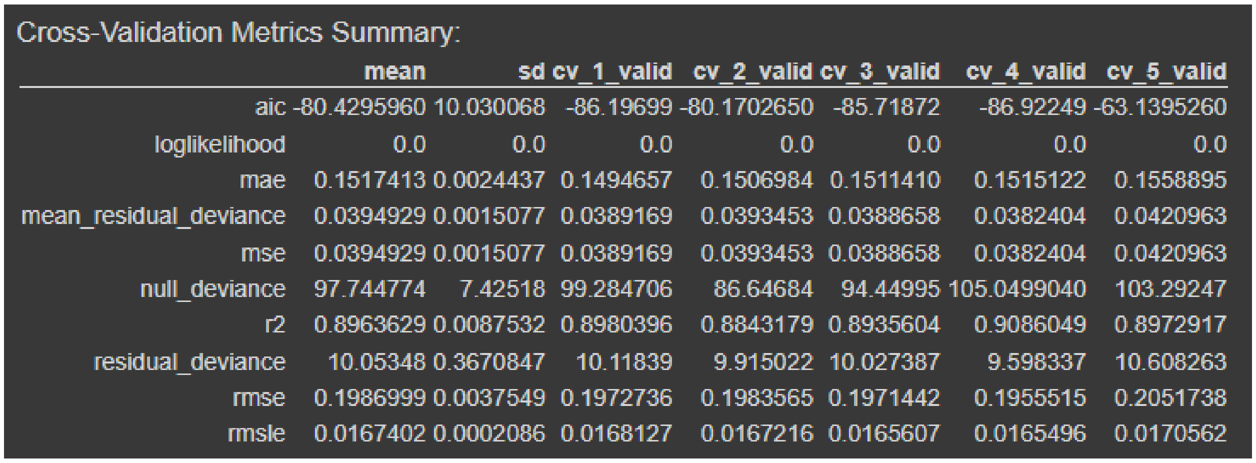
```
AutoML progress: |███████████████████████████████████████████████████████████████| (done) 100%

Model Details
=============
H2OStackedEnsembleEstimator : Stacked Ensemble
Model Key: StackedEnsemble_AllModels_1_AutoML_2_20240327_170426
```

Model Summary for Stacked Ensemble:

key	value
Stacking strategy	cross_validation
Number of base models (used / total)	8/10
# GBM base models (used / total)	3/4
# XGBoost base models (used / total)	3/3
# DRF base models (used / total)	1/2
# GLM base models (used / total)	1/1
Metalearner algorithm	GLM
Metalearner fold assignment scheme	Random
Metalearner nfolds	5
Metalearner fold_column	None
Custom metalearner hyperparameters	None



```
ModelMetricsRegressionGLM: stackedensemble
** Reported on train data. **

MSE: 0.012980660604468503
RMSE: 0.11393270208534731
MAE: 0.08702705512472024
RMSLE: 0.009575751625087722
Mean Residual Deviance: 0.012980660604468503
R^2: 0.966157356148678
Null degrees of freedom: 1272
Residual degrees of freedom: 1264
Null deviance: 488.2709820807015
Residual deviance: 16.524380949488403
AIC: -1897.6696157033157

ModelMetricsRegressionGLM: stackedensemble
** Reported on cross-validation data. **

MSE: 0.03960154318072879
RMSE: 0.19900136477102057
MAE: 0.1518181483156452
RMSLE: 0.01675609889005449
Mean Residual Deviance: 0.03960154318072879
R^2: 0.8967524872064533
Null degrees of freedom: 1272
Residual degrees of freedom: 1268
Null deviance: 488.72386898271805
Residual deviance: 50.412764469067746
AIC: -485.7558902214523
```



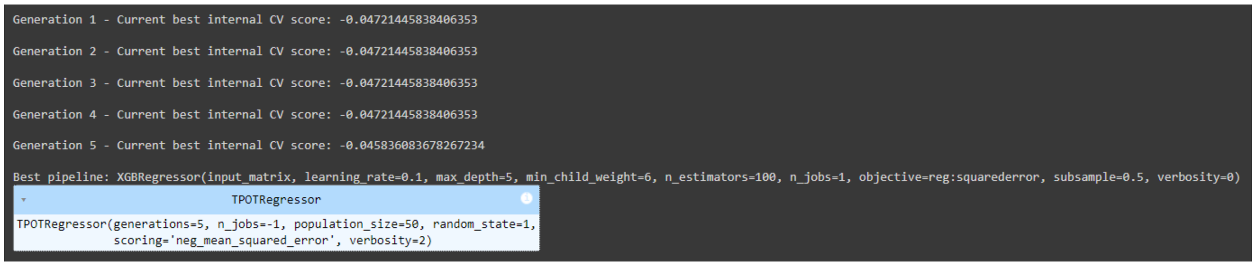
```
Cross-Validation Metrics Summary:
                          mean        sd  cv_1_valid  cv_2_valid  cv_3_valid   cv_4_valid  cv_5_valid
                aic -80.4295960 10.030068   -86.19699 -80.1702650   -85.71872    -86.92249 -63.1395260
      loglikelihood         0.0       0.0         0.0         0.0         0.0          0.0         0.0
                mae   0.1517413 0.0024437   0.1494657   0.1506984   0.1511410    0.1515122   0.1558895
mean_residual_deviance 0.0394929 0.0015077   0.0389169   0.0393453   0.0388658    0.0382404   0.0420963
                mse   0.0394929 0.0015077   0.0389169   0.0393453   0.0388658    0.0382404   0.0420963
      null_deviance   97.744774   7.42518   99.284706    86.64684    94.44995  105.0499040   103.29247
                 r2   0.8963629 0.0087532   0.8980396   0.8843179   0.8935604    0.9086049   0.8972917
  residual_deviance    10.05348 0.3670847    10.11839    9.915022   10.027387     9.598337   10.608263
               rmse   0.1986999 0.0037549   0.1972736   0.1983565   0.1971442    0.1955515   0.2051738
              rmsle   0.0167402 0.0002086   0.0168127   0.0167216   0.0165607    0.0165496   0.0170562
```

TPOT:

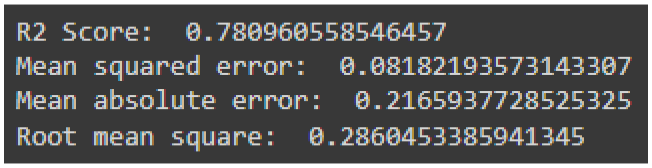
CODE:

```
!pip install tpot
import pandas as pd
data = pd.read_csv('/content/drive/MyDrive/Laptop - Sheet1.csv')
data.head()
data.tail()
data.isnull().sum()
data.replace('?',np.nan,inplace=True)
data.replace('#',np.nan,inplace=True)
data=pd.get_dummies(data)
data
X=data.drop('Price',axis=1)
X
y=data['Price']
y
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.2, random_state=32)
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(X_train,y_train)
from tpot import TPOTRegressor
model = TPOTRegressor(generations=5, population_size=50, cv=5,
scoring='neg_mean_squared_error', verbosity=2, random_state=1, n_jobs=-1)
model.fit(X_train, y_train)
y_pred=lr.predict(X_test)
from sklearn.metrics import
mean_squared_error,mean_absolute_error,r2_score,roc_auc_score,mean_squared_log_error
from sklearn.model_selection import cross_val_predict
print("R2 Score: ",r2_score(y_test,y_pred))
print("Mean squared error: ",mean_squared_error(y_test,y_pred))
print("Mean absolute error: ",mean_absolute_error(y_test,y_pred))
rms = np.sqrt(mean_squared_error(y_test,y_pred))
print("Root mean square: ",rms)
```

OUTPUT FOR TPOT:



```
Generation 1 - Current best internal CV score: -0.04721445838406353

Generation 2 - Current best internal CV score: -0.04721445838406353

Generation 3 - Current best internal CV score: -0.04721445838406353

Generation 4 - Current best internal CV score: -0.04721445838406353

Generation 5 - Current best internal CV score: -0.045836083678267234

Best pipeline: XGBRegressor(input_matrix, learning_rate=0.1, max_depth=5, min_child_weight=6, n_estimators=100, n_jobs=1, objective=reg:squarederror, subsample=0.5, verbosity=0)
TPOTRegressor
TPOTRegressor(generations=5, n_jobs=-1, population_size=50, random_state=1,
              scoring='neg_mean_squared_error', verbosity=2)
```



```
R2 Score:  0.780960558546457
Mean squared error:  0.08182193573143307
Mean absolute error:  0.2165937728525325
Root mean square:  0.2860453385941345
```

R2 SQUARE COMPARISON:

Manual: 0.780960558546457
H2o AutoML: 0.966157356148678
TPOT: 0.780960558546457

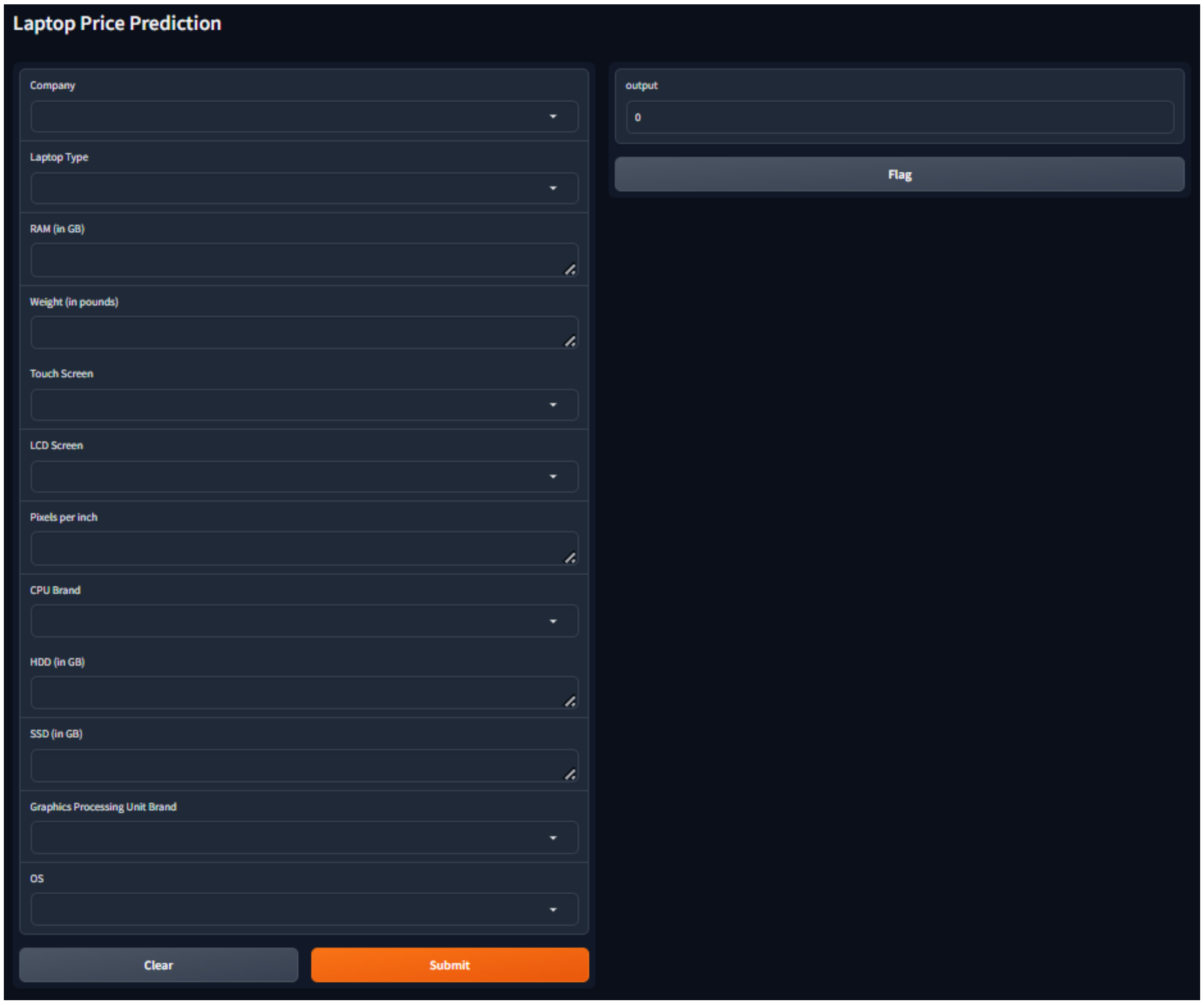
CONCLUSION:

The comparison of R2 square scores among different machine learning models for laptop price prediction offers valuable insights into their predictive performance. H2O AutoML emerges as the top performer with an impressively high R2 score of 0.966, indicating its superior ability to capture the variability in laptop prices. This robust performance suggests that H2O AutoML effectively leverages its automated machine learning capabilities to identify the most suitable algorithms and model configurations for the task at hand. In contrast, both the manual model and TPOT exhibit lower R2 scores of 0.781, indicating comparatively weaker predictive accuracy. While the manual model relies on human expertise and domain knowledge for feature selection and model tuning, TPOT automates the model selection process through genetic programming. Despite their efforts, they fail to match the predictive power demonstrated by H2O AutoML. Therefore, for tasks like laptop price prediction, where accuracy is paramount, adopting H2O AutoML would be prudent. Its ability to deliver highly accurate predictions not only enhances decision-making processes but also underscores the significance of automated machine learning tools in efficiently handling complex prediction tasks.

DEPLOYING THE MODEL USING GRADIO:

```
!pip install gradio
import gradio as gr

def predict(Company, TypeName, Ram, Weight, TouchScreen, Ips, Ppi, Cpu_brand, HDD, SSD,
Gpu_brand, Os):
    preprocessed_data = [Company, TypeName, Ram, Weight, int(TouchScreen), int(Ips), Ppi,
Cpu_brand, HDD, SSD, Gpu_brand, Os]
    preprocessed_data = np.array(preprocessed_data).reshape(1, -1)
    prediction = model.predict(preprocessed_data)
    predicted_price = round(prediction[0][0], 2)
    return [f"Predicted Laptop Price: ${predicted_price:.2f}"]

interface = gr.Interface(
    fn=predict,
    inputs=[
        gr.Dropdown(choices=["Acer", "Apple", "Asus", "Chuwi", "Dell", "Fujitsu", "Google",
"HP", "Huawei", "LG", "Lenovo", "MSI", "Mediacom", "Microsoft", "Razer", "Samsung",
"Toshiba", "Vero", "Xiaomi"], label="Company"),
        gr.Dropdown(choices=["2 in 1 Convertible", "Gaming", "Netbook", "Notebook",
"Ultrabook", "Workstation"], label="Laptop Type"),
        gr.Textbox(label="RAM (in GB)"),
        gr.Textbox(label="Weight (in pounds)"),
        gr.Dropdown(choices=[("1", "Yes"), ("0", "No")], label="Touch Screen"),
        gr.Dropdown(choices=[("1", "Yes"), ("0", "No")], label="LCD Screen"),
        gr.Textbox(label="Pixels per inch"),
        gr.Dropdown(choices=[("AMD Processor", "AMD"), ("Intel Core i3", "Intel"), ("Intel Core
i5", "Intel"), ("Intel Core i7", "Intel"), ("Other Core Processor", "Other")], label="CPU Brand"),
        gr.Textbox(label="HDD (in GB)"),
        gr.Textbox(label="SSD (in GB)"),
        gr.Dropdown(choices=[("AMD", "AMD"), ("Intel", "Intel"), ("Nvidia", "Nvidia")],
label="Graphics Processing Unit Brand"),
        gr.Dropdown(choices=[("Mac", "Apple"), ("Windows", "Windows"), ("Others", "Others")],
label="OS"),
    ],
    outputs=["number"],
    title="Laptop Price Prediction",
)
interface.launch()
```

INTERFACE AFTER DEPLOYMENT:


Laptop Price Prediction

Company

Laptop Type

RAM (in GB)

Weight (in pounds)

Touch Screen

LCD Screen

Pixels per inch

CPU Brand

HDD (in GB)

SSD (in GB)

Graphics Processing Unit Brand

OS

output

0