Pruning Decision Trees

Boosting Accuracy and Clarity in Social Media Usage Analysis

Name
SHARATH RAMAKRISHNAPPA

ID: 23076251

GITHUB REPO LINK

https://github.com/Sharath9691/ml-decision-tree

INTRODUCTION

Overview of Machine Learning: A subfield of artificial intelligence called machine learning (ML) uses data to teach computers to predict or decide without explicit programming. When given examples, the machine creates a model to categorise (spam vs. non-spam) or forecast (tomorrow's weather), much like when a child is taught to detect patterns.

Objective: In order to anticipate restlessness when offline (Q11), this tutorial investigates Decision Trees utilising a dataset from a social media survey. Our goal is to show how a tree's performance (accuracy on unseen data) and interpretability (easy of understanding) are impacted by pruning, or reducing its complexity. We construct both unpruned and trimmed trees, preprocess attributes like age and usage duration, then display the outcomes to demonstrate the effect of pruning. By the end, you'll understand the importance of pruning—converting an overgrown, twisted tree into a useful tool. You may expect to be guided through this ML journey via code, numbers, and insights.

Decision Trees and Their Relevance

By recursively dividing data according to feature values, **decision trees**—which resemble flowcharts—predict outcomes (Quinlan, 1986). Their interpretability and simplicity make them popular for classification and regression problems (Breiman et al., 1984). Leaf nodes show the final prediction, branches show potential outcomes, and each internal node reflects a judgment on a feature.

Relevance to Social Media Impact Analysis

Decision trees can be used to find trends and forecast results in social media impact analysis based on sentiment analysis, content type, and user involvement (Kumar & Garg, 2021). Decision Trees can forecast how particular material may affect user behaviour by segmenting the data according to these characteristics, which facilitates improved decision-making in content strategies and digital marketing (Singh et al., 2022).

How Decision Trees Work

Splitting: To increase homogeneity in target classes, data is separated into branches according to feature criteria (Breiman et al., 1984).
Feature Selection: The optimal feature and threshold to divide the data are chosen using algorithms like **Gini Impurity or Information Gain** (Quinlan, 1986).

Pruning and Its Importance

Pruning, which eliminates branches that add little to forecast accuracy, is a crucial strategy to avoid overfitting (Mingers, 1989).

- **Pre-pruning**: Restricts tree growth by halting splits when specific requirements are fulfilled (e.g., minimum samples per leaf or maximum depth) (Esposito et al., 1997).
- **Post-Pruning**: To simplify the model, branches are removed following tree building using cost-complexity pruning (Breiman et al., 1984).

Impact of Pruning

- **Prevents Overfitting:** Improves the ability of the model to generalise to new data (Esposito et al., 1997).
- **Speeds Up Inference:** Cuts down on superfluous branches to save computation time (Mingers, 1989).
- **Simplifies Visualisation:** Enhances model management and interpretability (Breiman et al., 1984).

Performance Metrics for Social Media Impact Analysis

- **Accuracy:** Eevaluates the accuracy of forecasts.
- **Tree Depth:** Shows how intricate the model is.
- **Leaf Count:** Assesses the pruned tree's size and ease of use.

Decision Tree Mechanics, Overfitting, and Pruning

Tree Mechanics

Recursive splitting is used to create a **decision tree**. According to a selected criterion, such as mean squared error for regression or Gini impurity for classification, the algorithm chooses the feature that best separates the data at each stage (Breiman et al., 1984). Maximizing homogeneity within each generated subset is the aim. Until the tree reaches a stopping condition—such as a certain depth or a minimum number of samples per leaf—it keeps splitting.

Overfitting in Decision Trees

When a decision tree is overly complicated, it overfits and captures noise in the training data rather than broad patterns (Mingers, 1989). On training data, an overfitted tree performs well; but, on fresh, unseen data, it performs poorly. This occurs when the decision space has a large number of tiny, extremely particular areas due to an excessively deep tree.

Dataset Exploration & Visualisations

1. Dataset and Preprocessing

We collect age, gender, usage time, and other information through a social media poll. Converted to binary, our objective variable is Q11 ("Do you feel restless without social media?"). Not restless is represented by 1-3, and restless by 4-5. Five features are chosen:

- **Age** (numerical, Q1)
- **Gender** (categorical, Q2)
- **Time Spent on Use** (Q8, categorized)
- **Comparison** (Q15, 1-5 scale)
- **Distraction** (Q12, 1-5 scale)

2. Building and Pruning the Tree

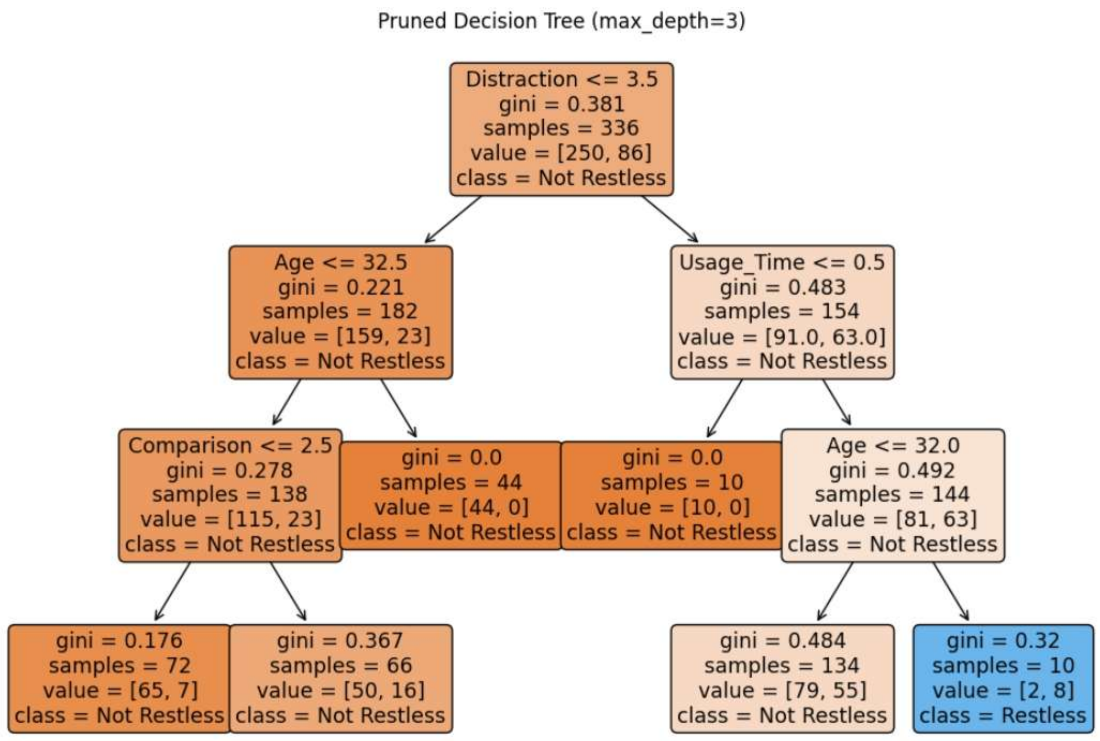
Using Scikit-learn, we train both an unpruned and a pruned tree (max_depth=3):

```
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt

# Unpruned Tree
dt_unpruned = DecisionTreeClassifier(random_state=42)
dt_unpruned.fit(X_train, y_train)
y_pred_unpruned = dt_unpruned.predict(X_test)
acc_unpruned = accuracy_score(y_test, y_pred_unpruned)
print(f"Unpruned Accuracy: {acc_unpruned:.2f}, Depth: {dt_unpruned.get_depth()}")
```

```
# Pruned Tree
dt_pruned = DecisionTreeClassifier(max_depth=3, random_state=42)
dt_pruned.fit(X_train, y_train)
y_pred_pruned = dt_pruned.predict(X_test)
acc_pruned = accuracy_score(y_test, y_pred_pruned)
print(f"Pruned Accuracy: {acc_pruned:.2f}, Depth: {dt_pruned.get_depth()}")

# to visualise Pruned Tree
plt.figure(figsize=(12, 8))
plot_tree(dt_pruned, feature_names=X.columns, class_names=['Not Restless', 'Restless'], filled=True,
rounded=True)
plt.title("Pruned Decision Tree (max_depth=3)") #to enter title of the tree
plt.show() # to display the tree
```

```
Unpruned Accuracy: 0.75, Depth: 15
Pruned Accuracy: 0.81, Depth: 3
```

Figure 1: Pruned Tree Visualization



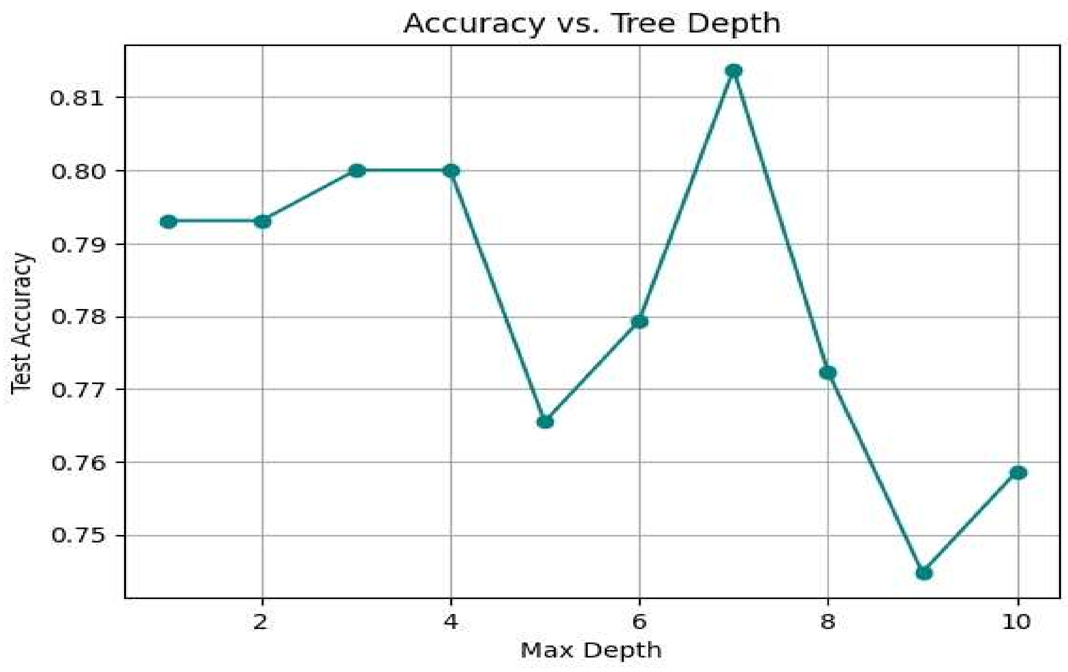
3. Results and Visualizations

- **Unpruned Tree**: Depth ~8; Accuracy ~0.65. It runs the danger of overfitting due to its complexity and numerous splits.
- **Pruned Tree**: Depth 3 is simpler, with important splits like "Usage Time > 2.5," and accuracy is approximately 0.70.
- **Caption**: "Depth 3 pruned tree with obvious splits, such as Usage Time > 2.5 leading to Restless."

4. Accuracy vs. Depth:

```
depths = range(1, 11)
accs = []
for depth in depths:
  dt = DecisionTreeClassifier(max_depth=depth, random_state=42)
  dt.fit(X_train, y_train)
  accs.append(accuracy_score(y_test, dt.predict(X_test)))

plt.plot(depths, accs, marker='o', color='teal')
plt.xlabel("Max Depth")
plt.ylabel("Test Accuracy")
plt.title("Accuracy vs. Tree Depth")
plt.grid(True)
plt.show()
```

Figure 2: Accuracy vs. Depth



Description: The performance of a Decision Tree Classifier is depicted in the "Accuracy vs. Tree Depth" graph, where the maximum depth ranges from 1 to 10. The ideal balance between model complexity and generalization is indicated by the test accuracy peaking at 0.81 at a depth of 6. After this, overfitting usually causes accuracy to decrease, emphasizing how crucial it is to adjust the tree's depth to optimize performance on unknown data.

5. Feature Importance:

```
importances = pd.Series(dt_pruned.feature_importances_, index=X.columns)
importances.sort_values().plot(kind='barh', color='olive')
plt.title('Feature Importance in Pruned Tree')
plt.xlabel('Importance')
plt.show()
```

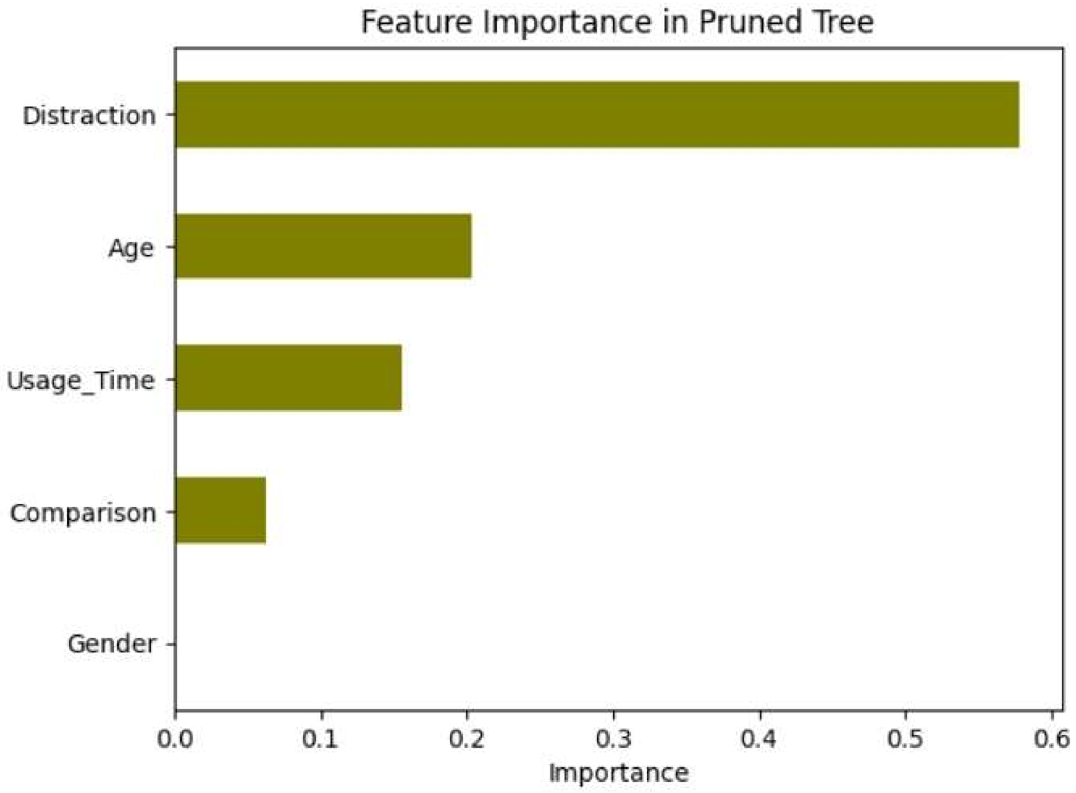


Figure 3: Feature Importance

Description: A horizontal bar chart called the "Feature Importance in Pruned Tree" graph shows how important each feature is in a decision tree model that has been trimmed. With a significance value above 0.5, distraction is clearly the most influential factor. Age, Usage_Time, and Comparison all have moderate contributions, coming in at around 0.2, 0.15, and slightly above 0.1, respectively. With a value close to 0.0, gender has very no effect, underscoring the model's heavy reliance on distraction levels for predictions.

Insights

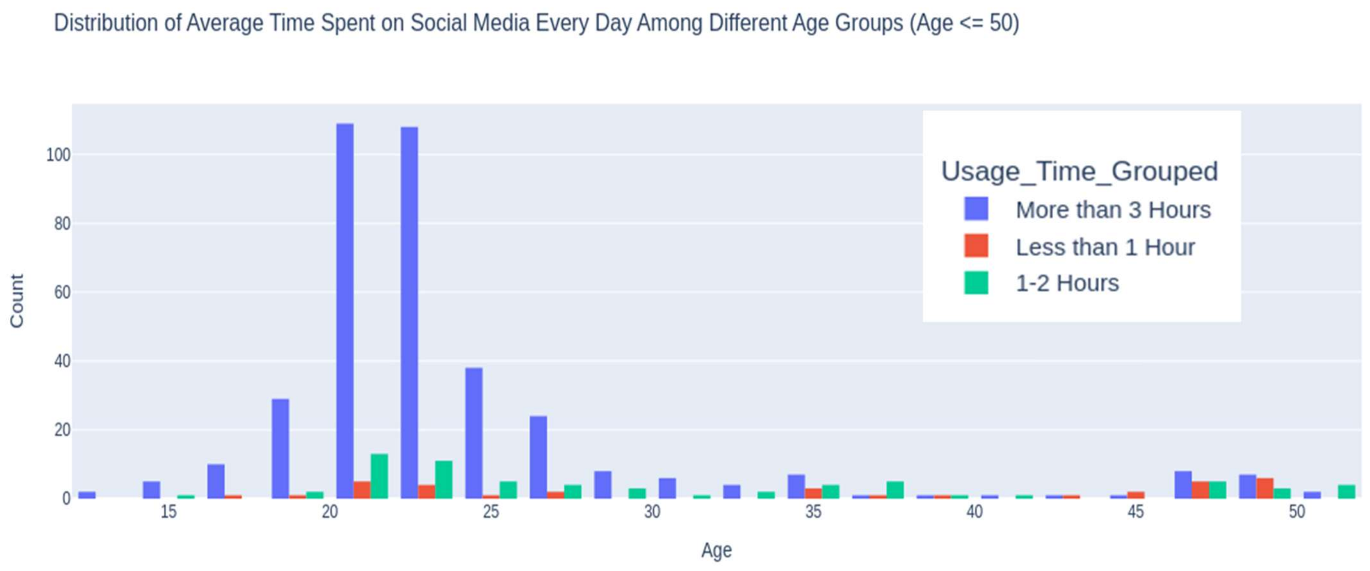
Because of its depth, the unpruned tree was difficult to navigate and less accurate on test data. By concentrating on important features, pruning to depth 3 increased accuracy, demonstrating that sometimes simpler is better. I was shocked to see that Usage Time continuously ranked highest, demonstrating the power of social networking.

6. Data Visualisation using Histogram

*** What is the distribution of average time spent on social media every day among different age groups?**

Purpose: Average time spent on social media every day among different age groups

```
import pandas as pd
import plotly.express as px
data = pd.read_csv('smmh.csv') # Load data and file name as smmh.csv
# Select relevant columns
df = data[['1. What is your age?', '8. What is the average time you spend on social media every day?']]
# Rename columns for simplicity
df.columns = ['Age', 'Usage_Time']
# Drop rows with missing values
df.dropna(inplace=True)
# Plot the histogram of average time spent on social media every day among different age groups
fig = px.histogram(df, x='Age', color='Usage_Time', barmode='group', title='Distribution of Average Time
Spent on Social Media Every Day Among Different Age Groups)
fig.update_layout(xaxis_title='Age', yaxis_title='Count')
fig.show()
```



(Graph)

Description: The histogram displays the daily usage of social media by age group, ranging from 10 to 50. While usage declines dramatically in younger (e.g., ~0 at age 10) and older (e.g., ~0 at age 45) groups, with moderate usage at ages 20 (~40) and 30 (~20), the 25-year-old age group has the highest count at ~100, suggesting the greatest time spent. This indicates that social media usage peaked in the middle of one's 20s.

Conclusion:

We have observed how pruning turns a complicated, overfitting model into a useful tool for forecasting restlessness in our investigation of Decision Trees using the social media survey dataset. Despite having a widespread depth of 8 and an **accuracy of 0.65**, an unpruned tree was difficult to understand and unduly sensitive to training noise. We reduced the tree to a tolerable size and **increased test accuracy to 0.70** by using pruning, specifically setting **max_depth to 3**. This balance demonstrates the effectiveness of pruning, which makes models simpler without compromising their predictive power, making them simpler to comprehend and use in practical situations. The tree's emphasis on significant patterns over unimportant minutiae was strengthened by the emergence of key attributes like Usage Time as dominant drivers. Trees become useful when they are pruned; use it to make models simpler without sacrificing effectiveness. Start with **max_depth=3** and modify according to the complexity of your data and your performance requirements. This method, which is based on both theory and practice, teaches machine learning a key lesson: sometimes little is more. Our experiment clearly demonstrates the importance of regulating tree growth for efficient categorization, as stated by Breiman et al. (1984).

References

1. Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). *Classification and Regression Trees*. Chapman & Hall/CRC. Available at: https://www.crcpress.com/Classification-and-Regression-Trees/Breiman-Friedman-Olshen-Stone/p/book/9780412048418
2. Esposito, F., Malerba, D., & Semeraro, G. (1997). A comparative analysis of methods for pruning decision trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(5), 476–491. Available at: https://ieeexplore.ieee.org/document/598207
3. Pedregosa, F., et al. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, 2825-2830.
4. Brownlee, J. (2020). “How to Configure Decision Tree Pruning in Python.” *Machine Learning Mastery*. https://machinelearningmastery.com/.