Muhammad Shafeen

22P-9278

BS-AI

ML

In [1]:

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from itertools import combinations
import matplotlib.pyplot as plt
```

In [2]:

```
data = {
    'Color': ['Red', 'Blue', 'Green', 'Black', 'Blue', 'Red', 'Green', 'Black',
    'Size': ['Full', 'Medium', 'Standard', 'Full', 'Standard', 'Medium', 'Full',
    'Model': ['G15 5510', 'G15 5515', 'G15 5520', 'G15 5510', 'G15 5515', 'G15 5
    'Material': ['Plastic', 'Metal', 'Plastic', 'Plastic', 'Metal', 'Plastic', '
    'Target': [1, 1, 1, 0, 1, 0, 0, 1, 0, 1]
}
```

In [3]:

```
df = pd.DataFrame(data)
df
```

Out[3]:

	Color	Size	Model	Material	Target
0	Red	Full	G15 5510	Plastic	1
1	Blue	Medium	G15 5515	Metal	1
2	Green	Standard	G15 5520	Plastic	1
3	Black	Full	G15 5510	Plastic	0
4	Blue	Standard	G15 5515	Metal	1
5	Red	Medium	G15 5510	Plastic	0
6	Green	Full	G15 5515	Metal	0
7	Black	Medium	G15 5520	Plastic	1
8	Red	Standard	G15 5510	Plastic	0
9	Green	Full	G15 5520	Plastic	1

In [4]:

```
def evaluate_feature_combination(features):
    X = pd.get_dummies(df[features])
    X_train, X_test, y_train, y_test = train_test_split(X, df['Target'], test_si
    clf = DecisionTreeClassifier(random_state=42)
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    return accuracy_score(y_test, y_pred), clf
```

In [5]:

```
columns = df.drop(columns=['Target']).columns
```

In [6]:

```
results = []
for r in range(1, len(columns) + 1):
    for combo in combinations(columns, r):
        accuracy, clf = evaluate_feature_combination(list(combo))
        results.append({'Features': combo, 'Accuracy': accuracy, 'Model': clf})
```

In [7]:

```
results_df = pd.DataFrame(results)
results_df
```

Out[7]:

	Features	Accuracy	Model
0	(Color,)	0.333333	DecisionTreeClassifier(random_state=42)
1	(Size,)	0.333333	DecisionTreeClassifier(random_state=42)
2	(Model,)	0.666667	DecisionTreeClassifier(random_state=42)
3	(Material,)	0.000000	DecisionTreeClassifier(random_state=42)
4	(Color, Size)	0.333333	DecisionTreeClassifier(random_state=42)
5	(Color, Model)	0.333333	DecisionTreeClassifier(random_state=42)
6	(Color, Material)	0.333333	DecisionTreeClassifier(random_state=42)
7	(Size, Model)	0.333333	DecisionTreeClassifier(random_state=42)
8	(Size, Material)	0.333333	DecisionTreeClassifier(random_state=42)
9	(Model, Material)	0.666667	DecisionTreeClassifier(random_state=42)
10	(Color, Size, Model)	0.333333	DecisionTreeClassifier(random_state=42)
11	(Color, Size, Material)	0.333333	DecisionTreeClassifier(random_state=42)
12	(Color, Model, Material)	0.333333	DecisionTreeClassifier(random_state=42)
13	(Size, Model, Material)	0.333333	DecisionTreeClassifier(random_state=42)
14	(Color, Size, Model, Material)	0.333333	DecisionTreeClassifier(random_state=42)

In [8]:

```
best_result = results_df.loc[results_df['Accuracy'].idxmax()]
worst_result = results_df.loc[results_df['Accuracy'].idxmin()]
average_accuracy = results_df['Accuracy'].mean()
```

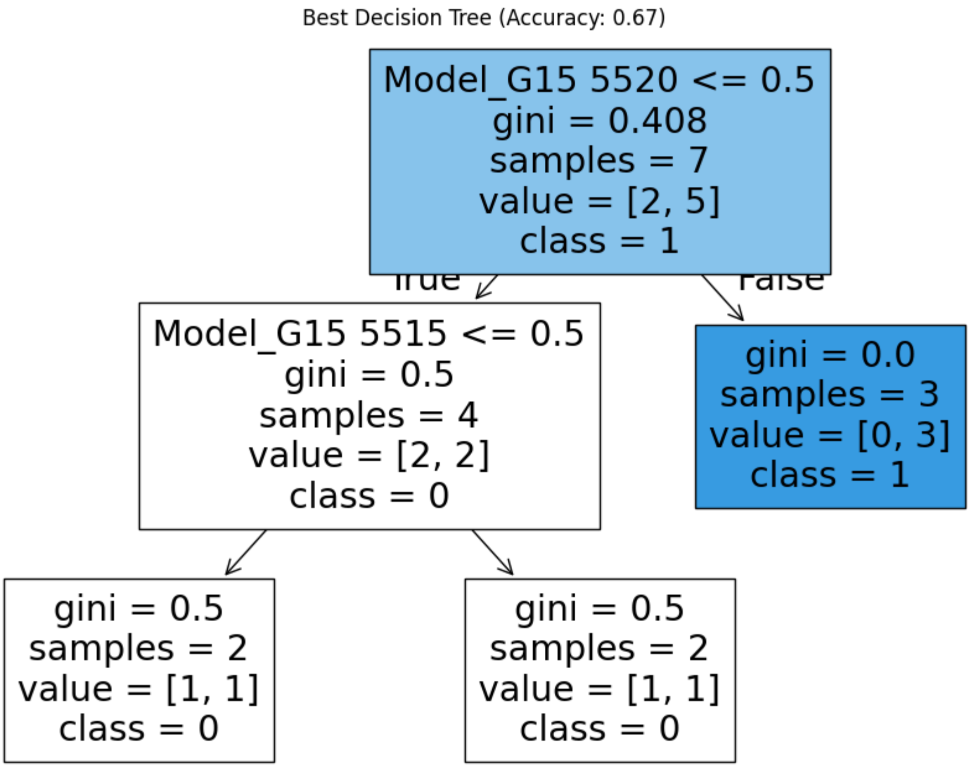
In [9]:

```
print(f"Best Combination: {best_result['Features']}, Accuracy: {best_result['Acc
print(f"Worst Combination: {worst_result['Features']}, Accuracy: {worst_result['
print(f"Average Accuracy: {average_accuracy:.2f}")
```

```
Best Combination: ('Model',), Accuracy: 0.67
Worst Combination: ('Material',), Accuracy: 0.00
Average Accuracy: 0.36
```

In [10]:

```
X_best = pd.get_dummies(df[list(best_result['Features'])])
plt.figure(figsize=(12, 8))
plot_tree(best_result['Model'], feature_names=X_best.columns, class_names=['0',
plt.title(f'Best Decision Tree (Accuracy: {best_result["Accuracy"]:.2f})')
plt.show()
```

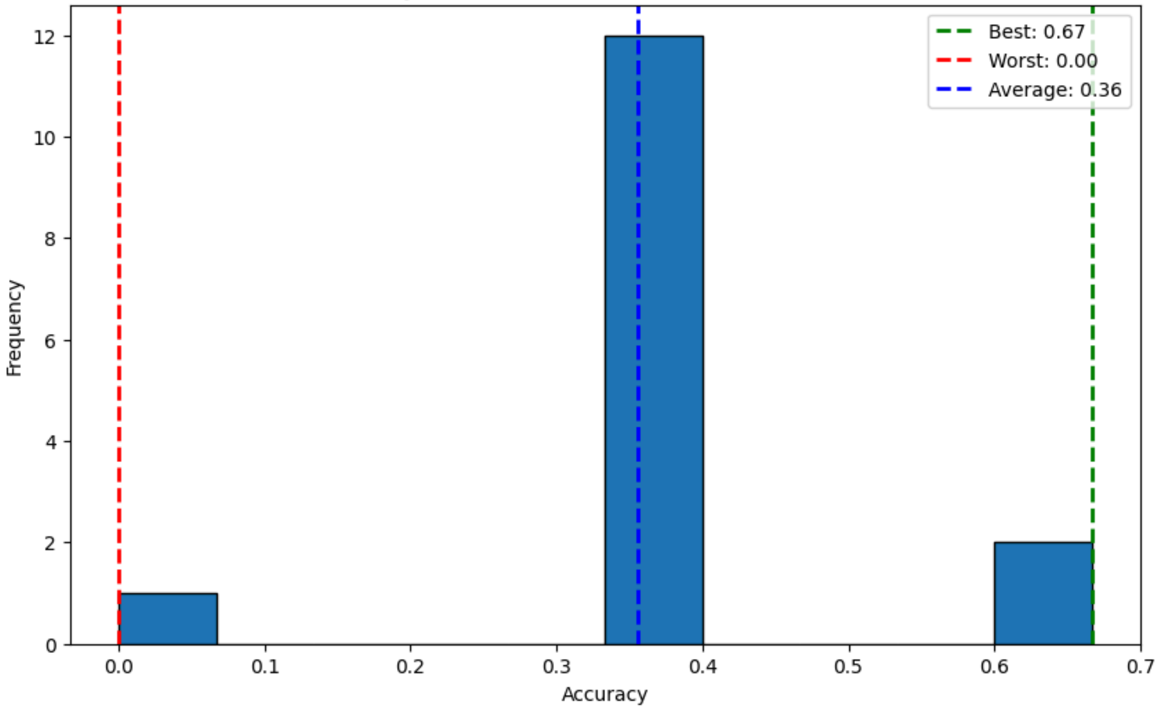


In [11]:

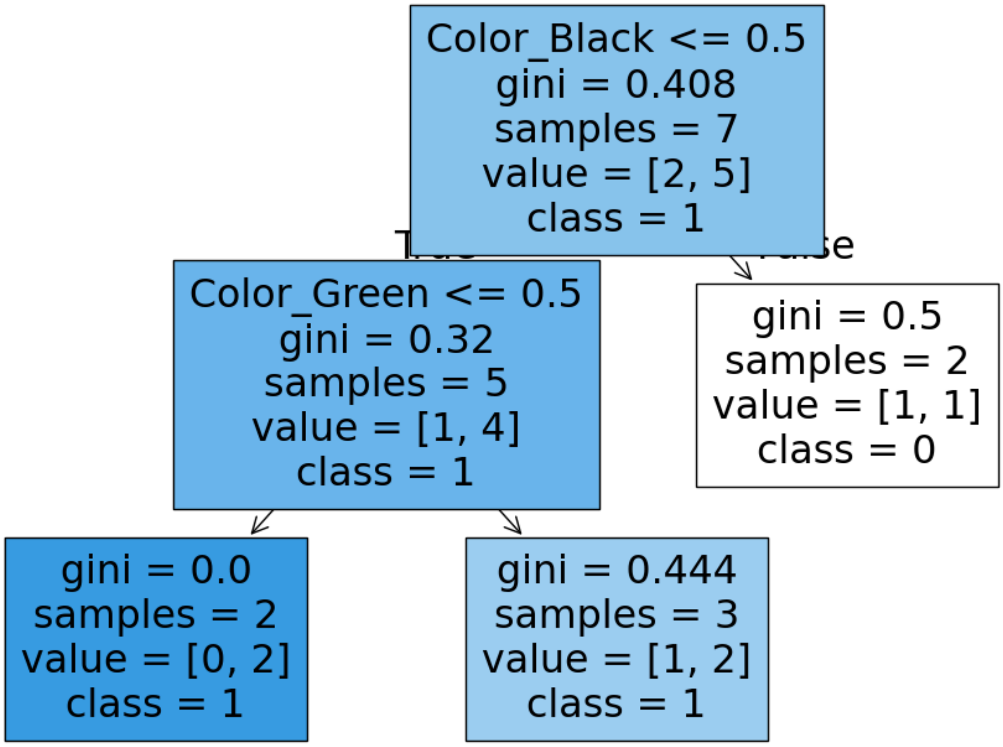
```
plt.figure(figsize=(10, 6))
plt.hist(results_df['Accuracy'], bins=10, edgecolor='black')
plt.axvline(best_result['Accuracy'], color='g', linestyle='dashed', linewidth=2,
plt.axvline(worst_result['Accuracy'], color='r', linestyle='dashed', linewidth=2
plt.axvline(average_accuracy, color='b', linestyle='dashed', linewidth=2, label=
plt.title('Accuracy Distribution Across Feature Combinations')
plt.xlabel('Accuracy')
plt.ylabel('Frequency')
plt.legend()
plt.show()


for index, row in results_df.iterrows():
    plt.figure(figsize=(12, 8))
    X_combo = pd.get_dummies(df[list(row['Features'])])
    plot_tree(row['Model'], feature_names=X_combo.columns, class_names=['0', '1'
    plt.title(f"Decision Tree for Features: {row['Features']} (Accuracy: {row['A
    plt.show()
```

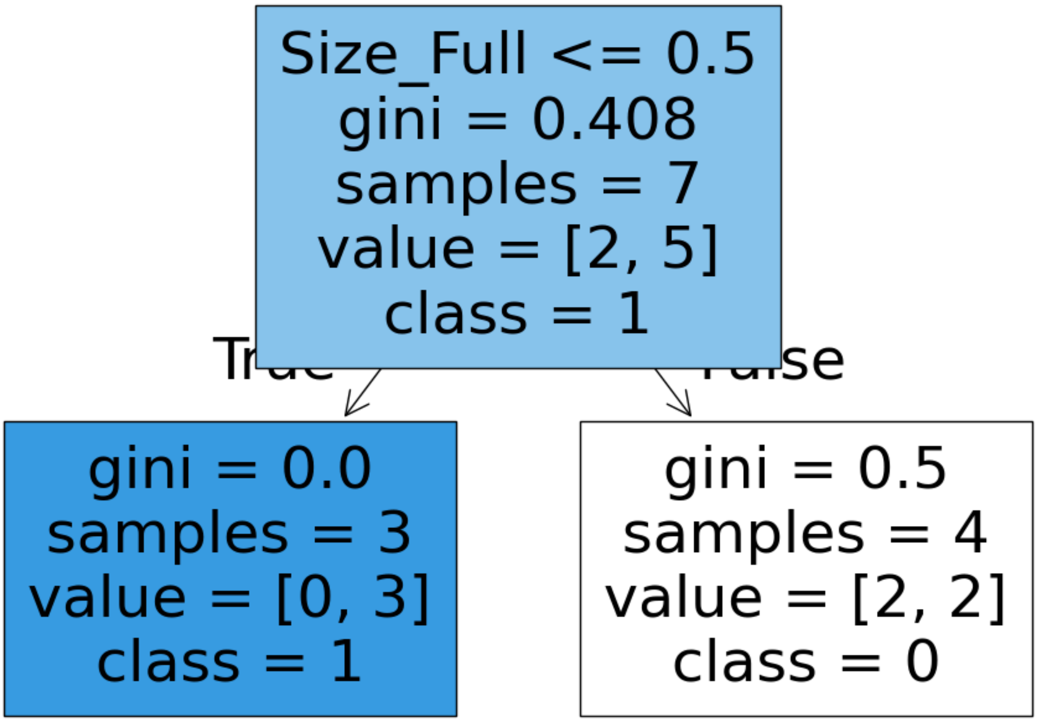
Accuracy Distribution Across Feature Combinations



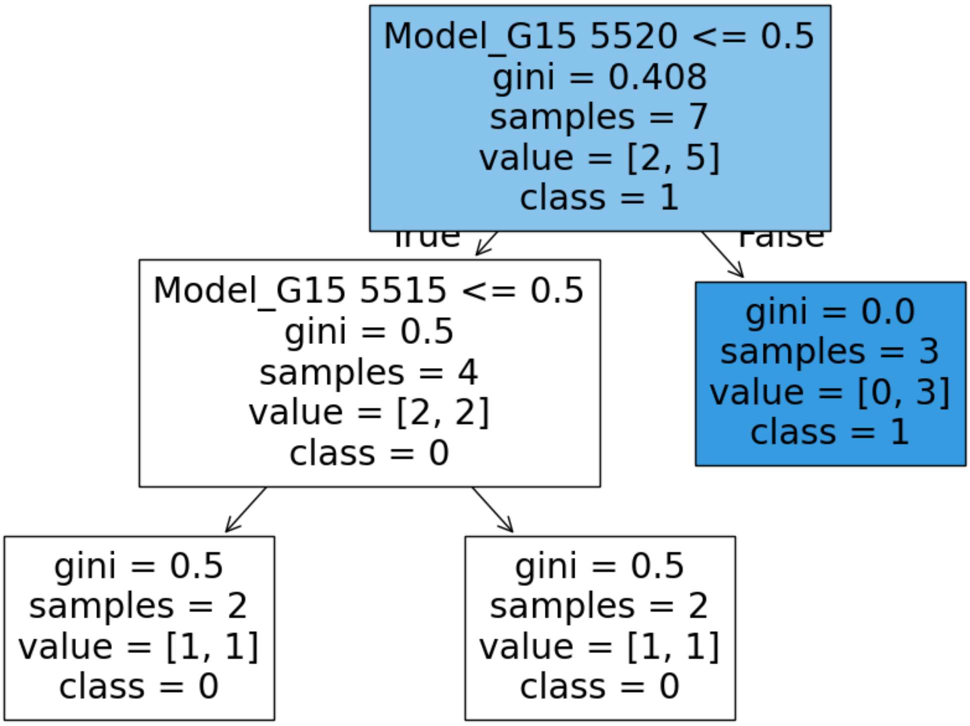
Decision Tree for Features: ('Color',) (Accuracy: 0.33)

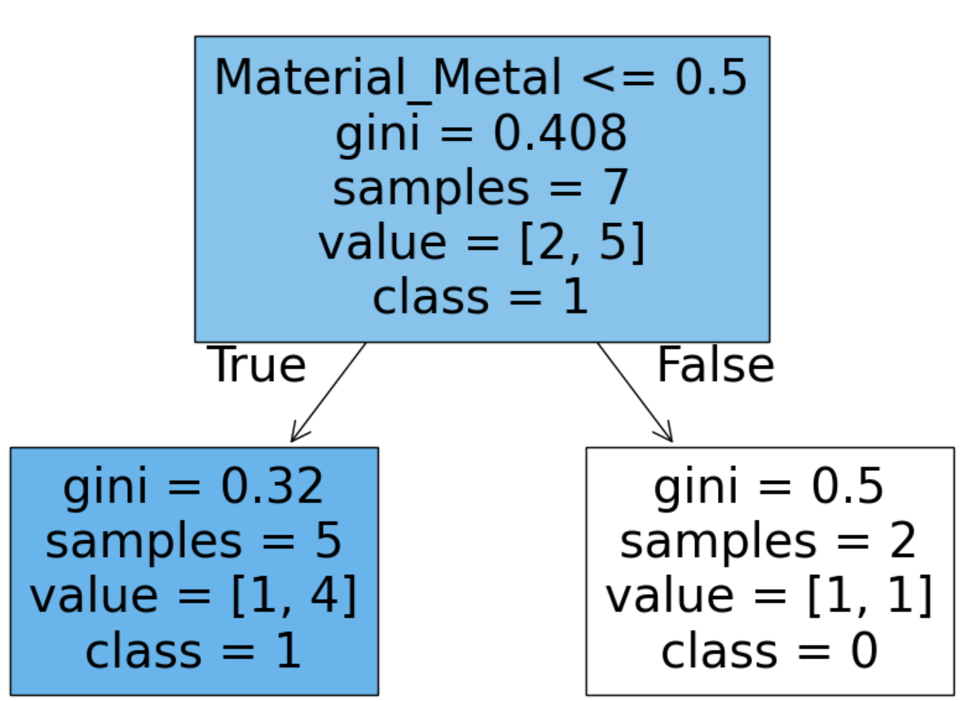


Decision Tree for Features: ('Size',) (Accuracy: 0.33)

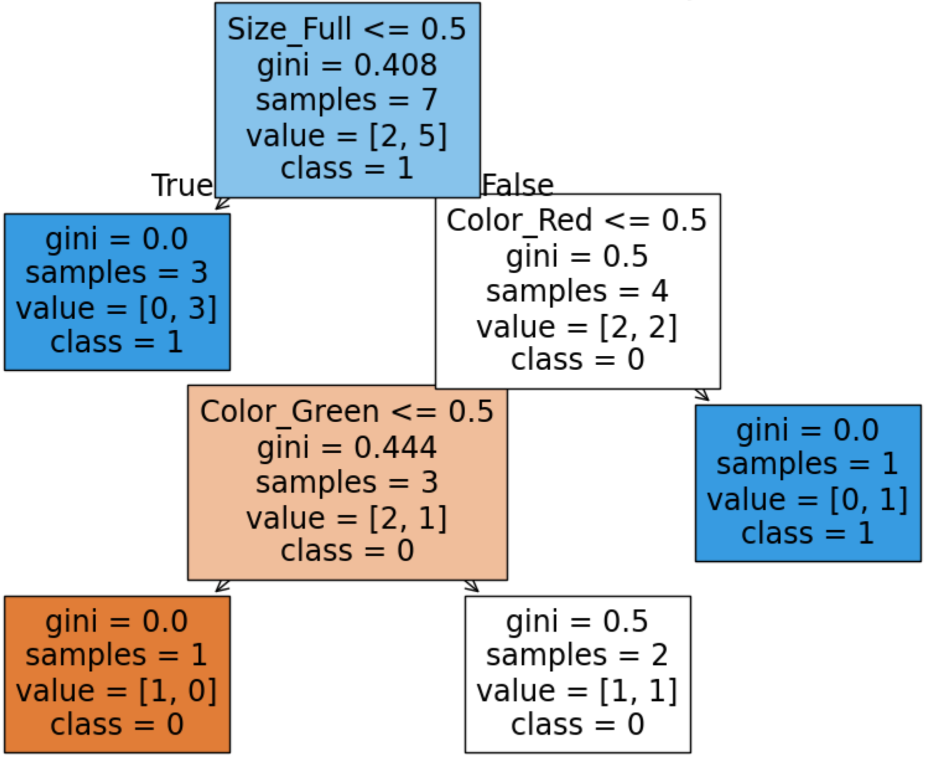


Decision Tree for Features: ('Model',) (Accuracy: 0.67)



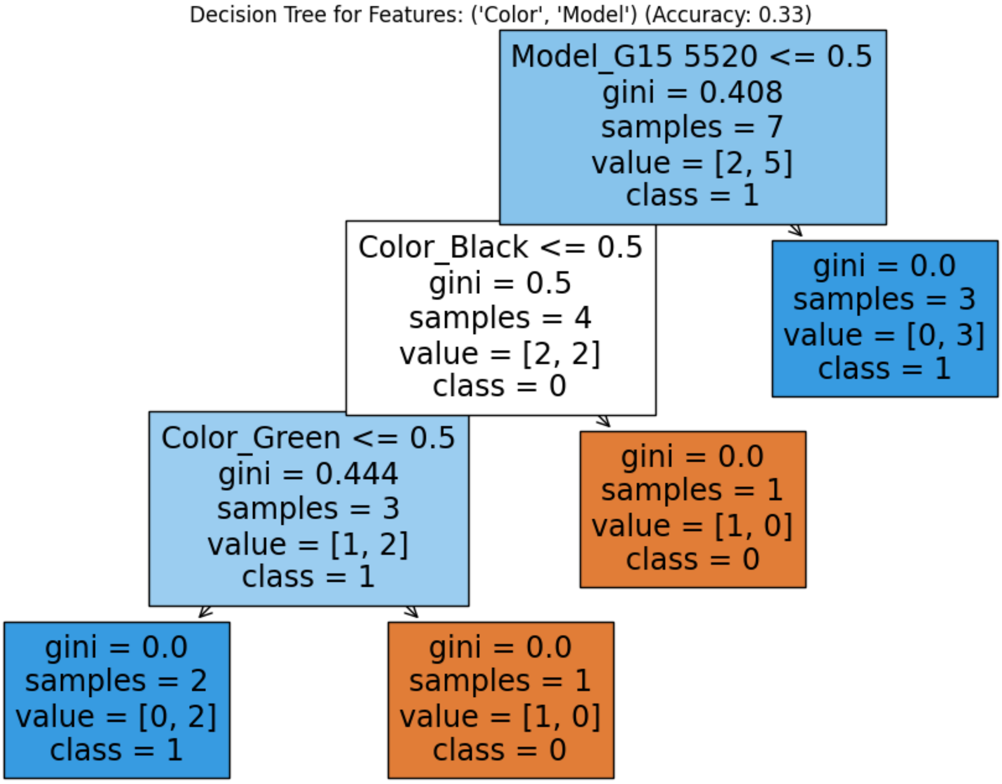


Decision Tree for Features: ('Material',) (Accuracy: 0.00)

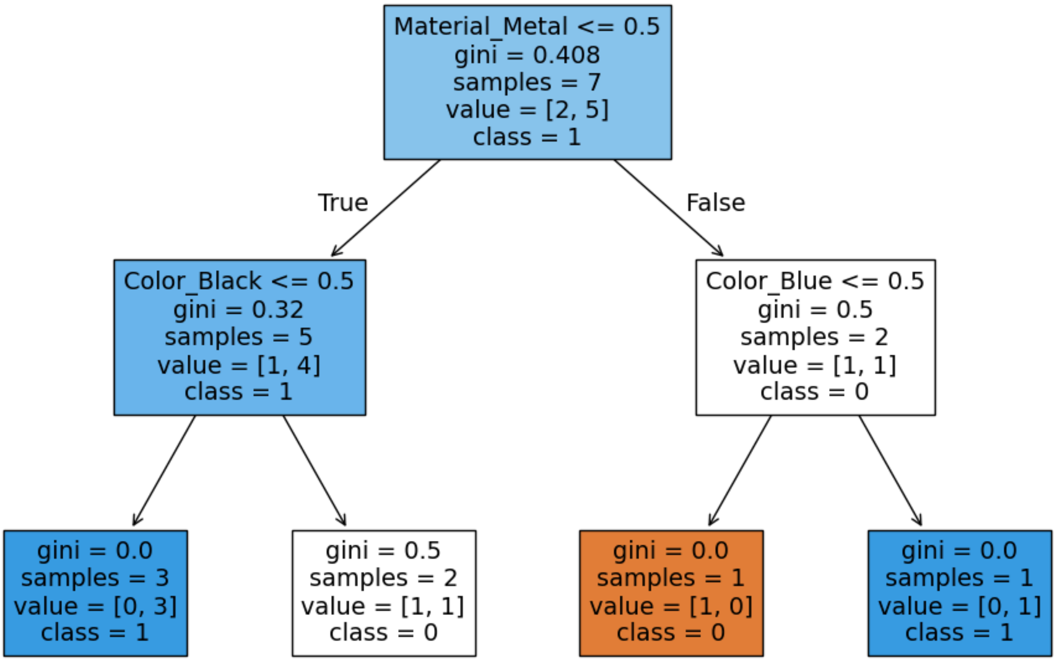


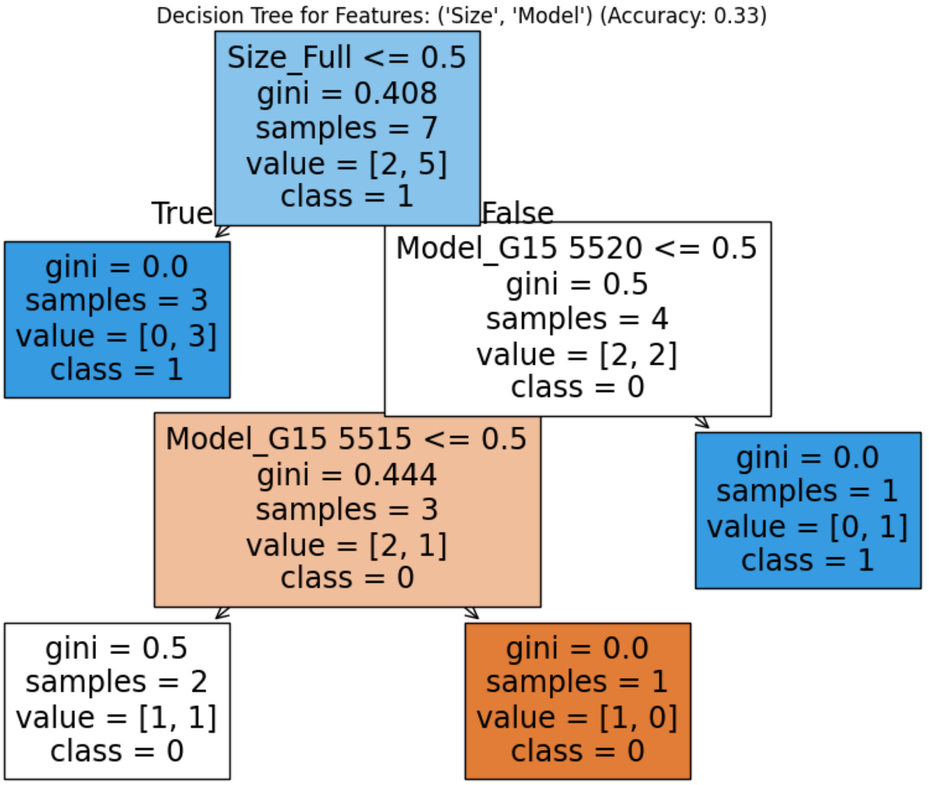
Decision Tree for Features: ('Color', 'Size') (Accuracy: 0.33)

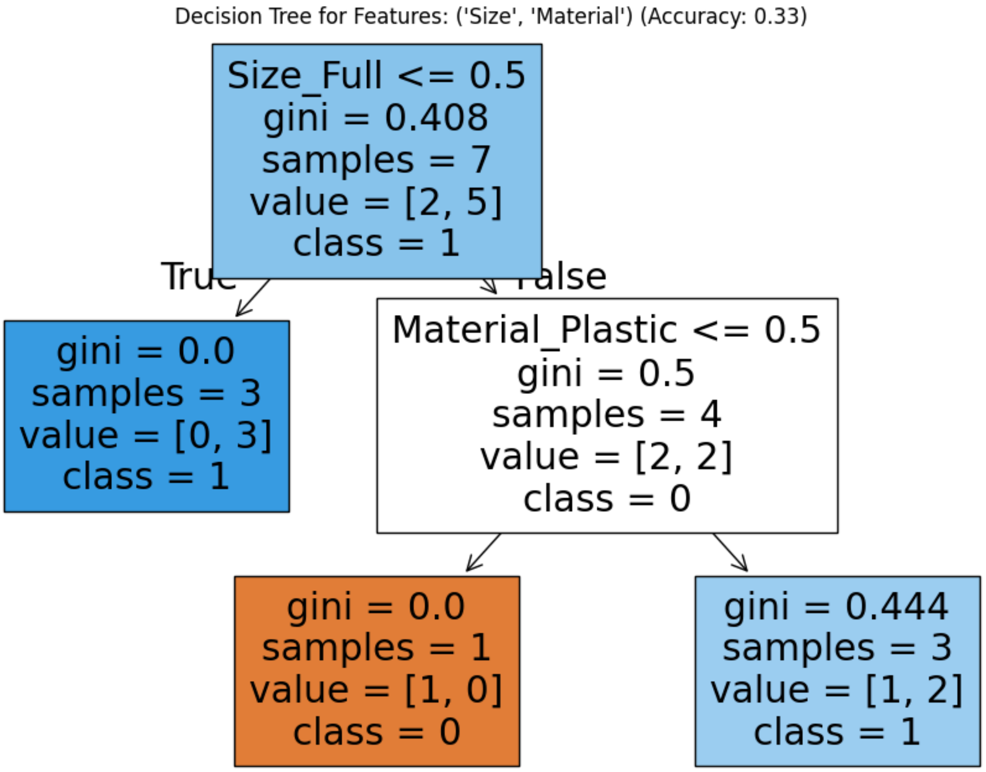
Decision Tree for Features: ('Color', 'Model') (Accuracy: 0.33)



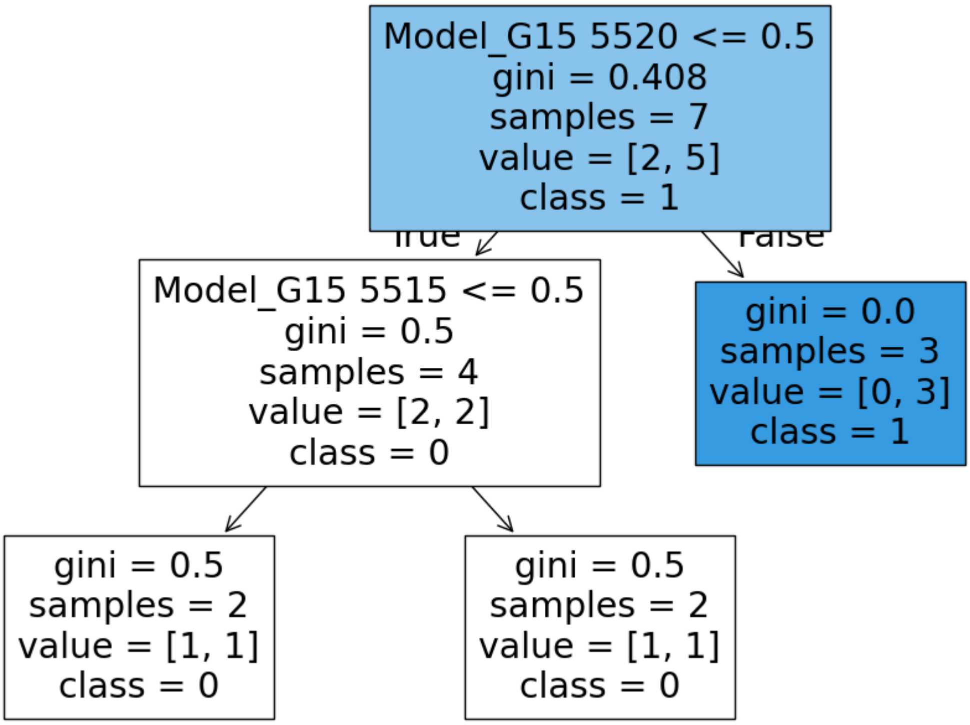
Decision Tree for Features: ('Color', 'Material') (Accuracy: 0.33)



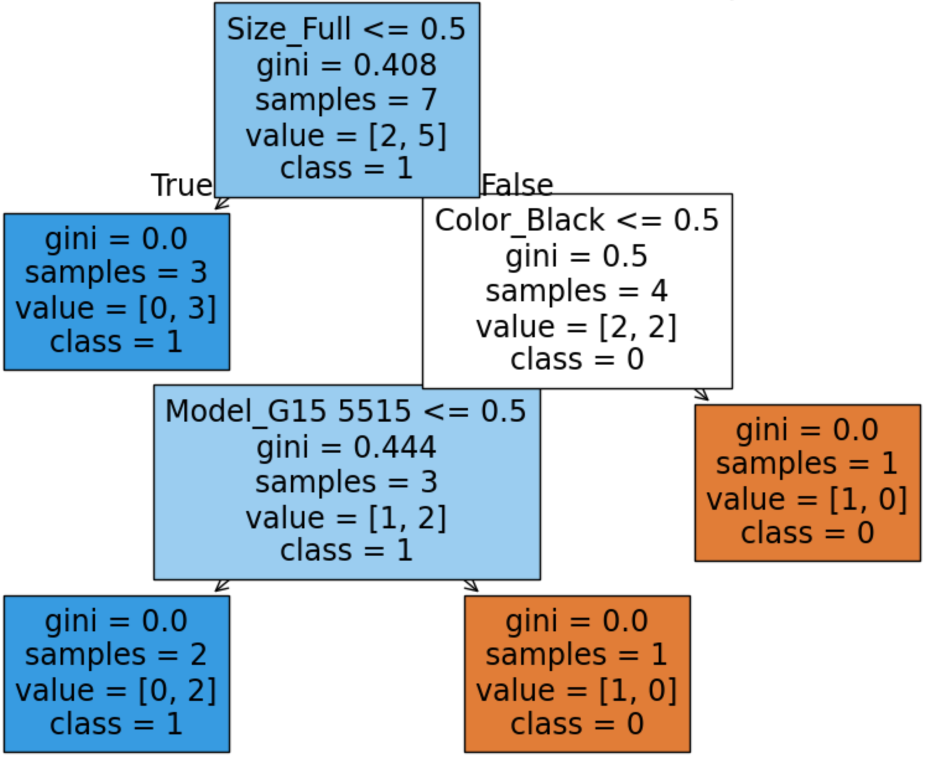

Decision Tree for Features: ('Size', 'Model') (Accuracy: 0.33)
Size_Full <= 0.5
gini = 0.408
samples = 7
value = [2, 5]
class = 1
True
False
gini = 0.0
samples = 3
value = [0, 3]
class = 1
Model_G15 5520 <= 0.5
gini = 0.5
samples = 4
value = [2, 2]
class = 0
Model_G15 5515 <= 0.5
gini = 0.444
samples = 3
value = [2, 1]
class = 0
gini = 0.0
samples = 1
value = [0, 1]
class = 1
gini = 0.5
samples = 2
value = [1, 1]
class = 0
gini = 0.0
samples = 1
value = [1, 0]
class = 0


Decision Tree for Features: ('Size', 'Material') (Accuracy: 0.33)
Size_Full <= 0.5
gini = 0.408
samples = 7
value = [2, 5]
class = 1
True
False
gini = 0.0
samples = 3
value = [0, 3]
class = 1
Material_Plastic <= 0.5
gini = 0.5
samples = 4
value = [2, 2]
class = 0
gini = 0.0
samples = 1
value = [1, 0]
class = 0
gini = 0.444
samples = 3
value = [1, 2]
class = 1

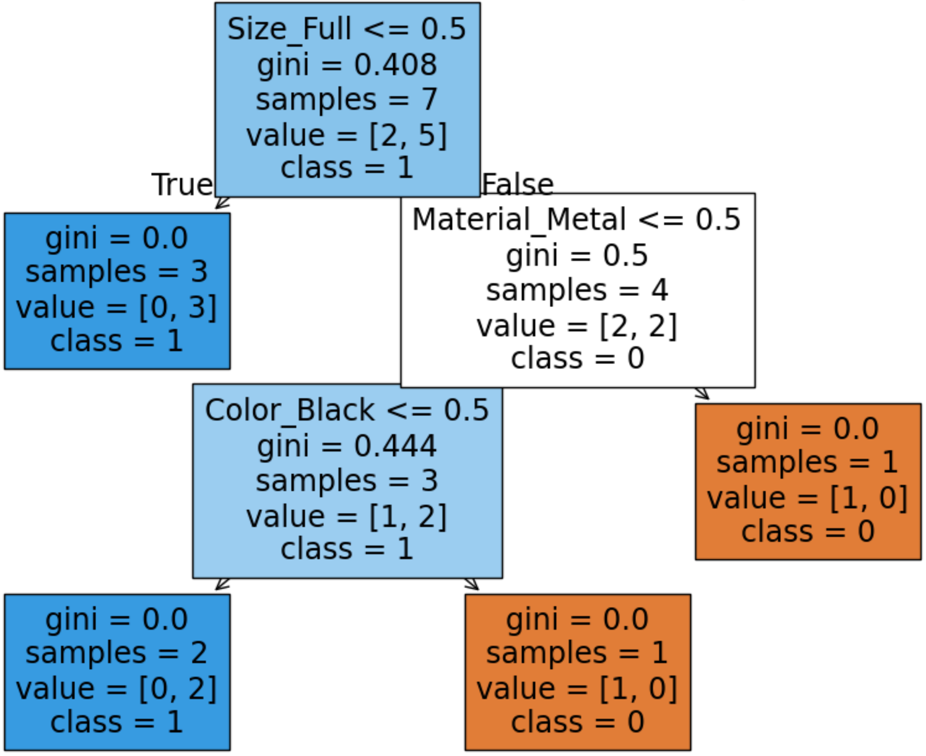
Decision Tree for Features: ('Model', 'Material') (Accuracy: 0.67)



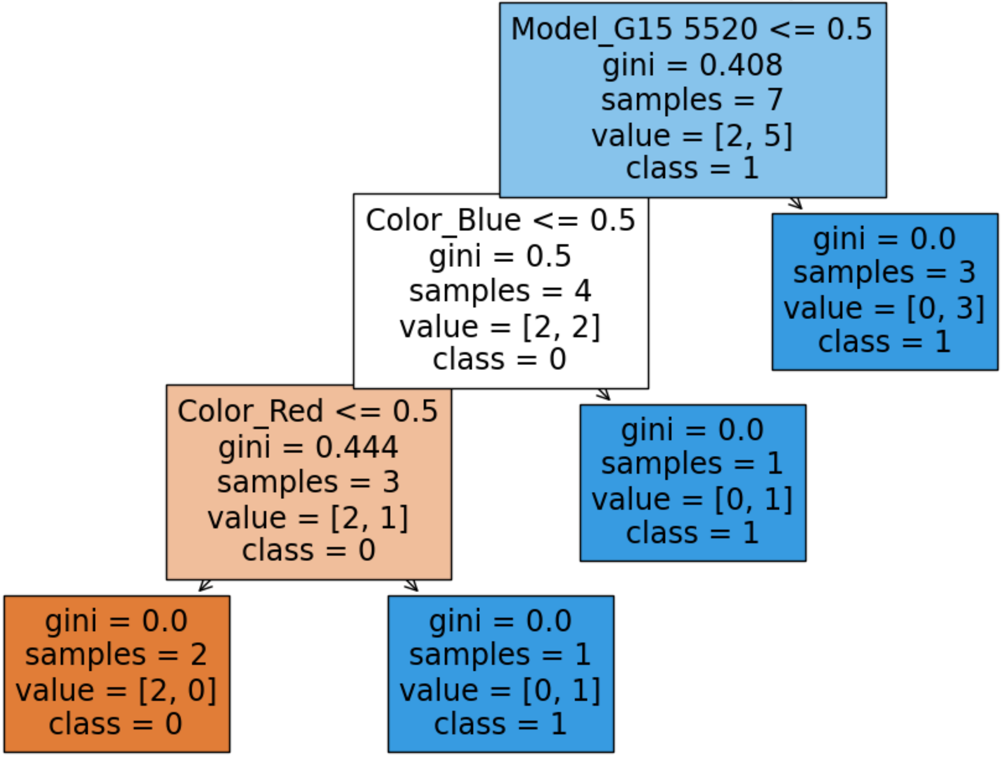
Decision Tree for Features: ('Color', 'Size', 'Model') (Accuracy: 0.33)

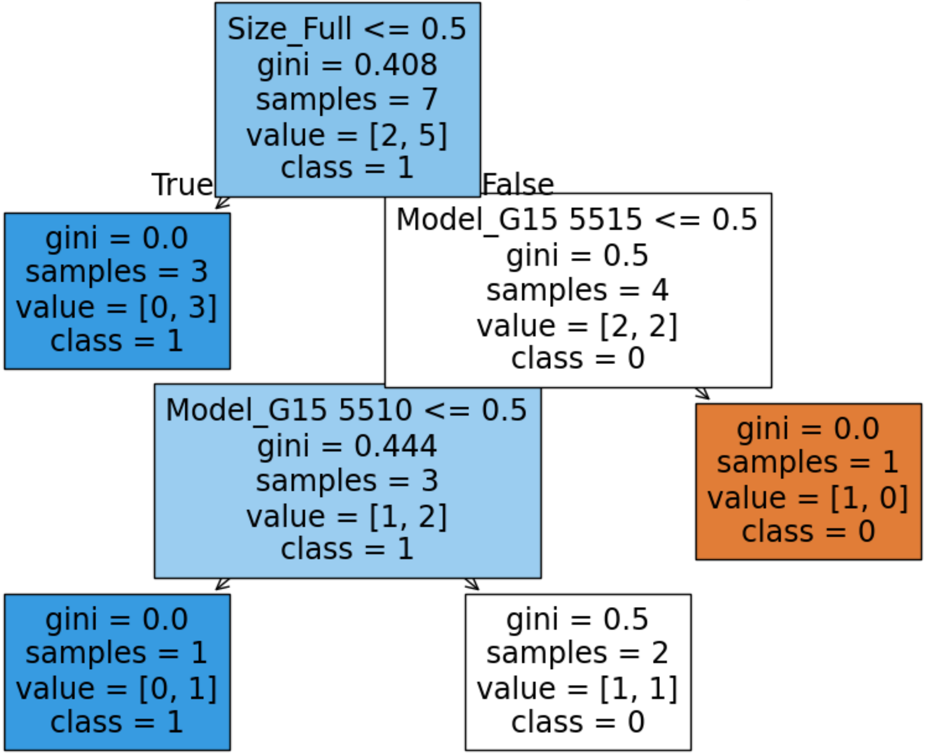


Decision Tree for Features: ('Color', 'Size', 'Material') (Accuracy: 0.33)

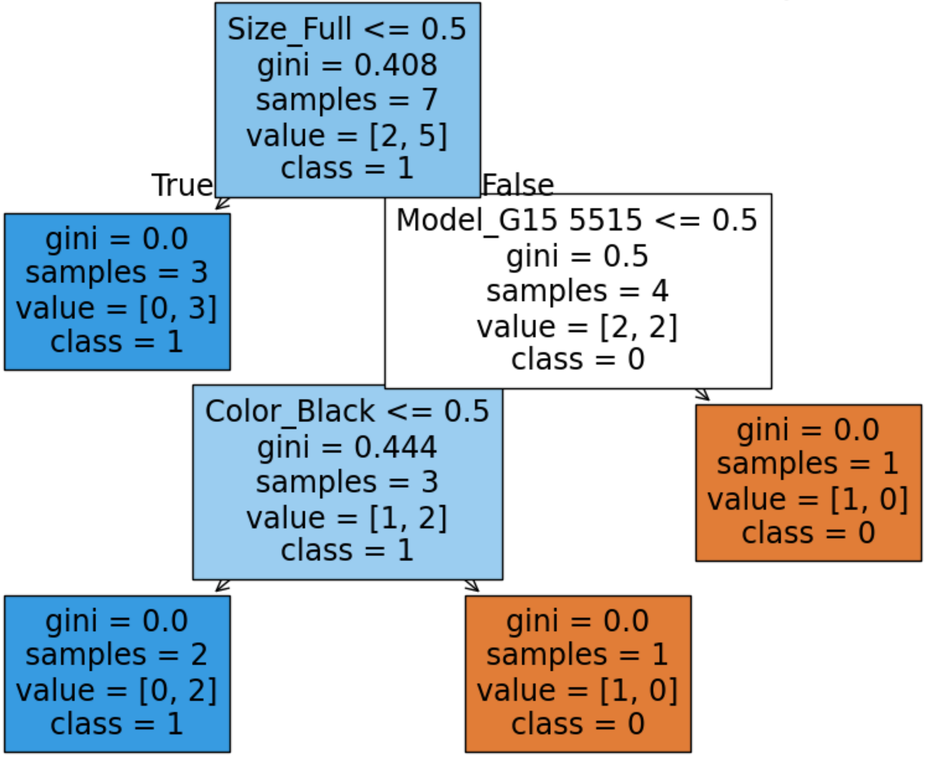


Decision Tree for Features: ('Color', 'Model', 'Material') (Accuracy: 0.33)





Decision Tree for Features: ('Size', 'Model', 'Material') (Accuracy: 0.33)



Decision Tree for Features: ('Color', 'Size', 'Model', 'Material') (Accuracy: 0.33)