PREDICTIVE ANALYSIS WITH DECISION TREES

Submitted by:

Name : Sahana S

Register Number : 727723EUCS200

Class : CSE – ‘C’

Year : III

Department : Computer Science and Engineering

1. Introduction

Predictive analysis involves analyzing historical data to make predictions about future outcomes.Decision Trees are popular machine learning models used for predictive analytics due to their simplicity, interpretability, and ability to handle both classification and regression problems.

2. Objectives

- Implement Decision Tree for classification
- Understand entropy and Gini index
- Apply pruning techniques to avoid overfitting
- Visualize the decision tree
- Evaluate feature importance

3. Dataset Description

The Heart Disease dataset consists of patient medical records used to predict the presence of heart disease. The dataset contains numerical and categorical attributes related to heart health.

4. Decision Tree Theory

4.1 Entropy

Entropy measures the level of uncertainty or impurity in a dataset. Lower entropy indicates purer data.

$$Entropy(S) = -\sum p_i \log_2(p_i)$$

Where p_i is the probability of class i.

4.2 Information Gain

Used to decide the best feature to split.

$$IG(S, A) = Entropy(S) - \sum \frac{|S_v|}{|S|} Entropy(S_v)$$

4.3 Gini Index

Measures impurity (used by CART algorithm).

$$Gini = 1 - \sum p_i^2$$

Lower Gini → better split.

5. Overfitting in Decision Trees

Decision trees tend to memorize training data leading to overfitting.

Solutions:

- **Pre-Pruning:** Stop tree growth early
- **Post-Pruning:** Remove unnecessary branches

6. Methodology

1. Load dataset
2. Preprocess data
3. Compute entropy and Gini manually
4. Train Decision Tree
5. Apply pruning
6. Evaluate performance

7. Visualize tree
8. Interpret feature importance

7. Tools & Technologies

- Python
- NumPy
- Pandas
- Scikit-learn
- Matplotlib
- Graphviz

8. Results

- Accuracy before pruning
- Accuracy after pruning
- Improved generalization
- Key features identified

9. Conclusion

Decision Trees are powerful and interpretable machine learning models. By using entropy, Gini index, and pruning techniques, model performance can be optimized and overfitting can be avoided.

10. Future Enhancements

- Use Random Forests
- Apply Gradient Boosting
- Hyperparameter tuning

Source Code:

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt


# 1. IMPORT LIBRARIES

from sklearn.model_selection import train_test_split

from sklearn.tree import DecisionTreeClassifier, plot_tree

from sklearn.metrics import accuracy_score, classification_report

from sklearn.preprocessing import LabelEncoder


# 2. LOAD DATASET

df = pd.read_csv("heart.csv")


print("\nDataset Preview:")

display(df.head())

print("\nDataset Shape:", df.shape)


#  FIX: HANDLE CATEGORICAL DATA
```

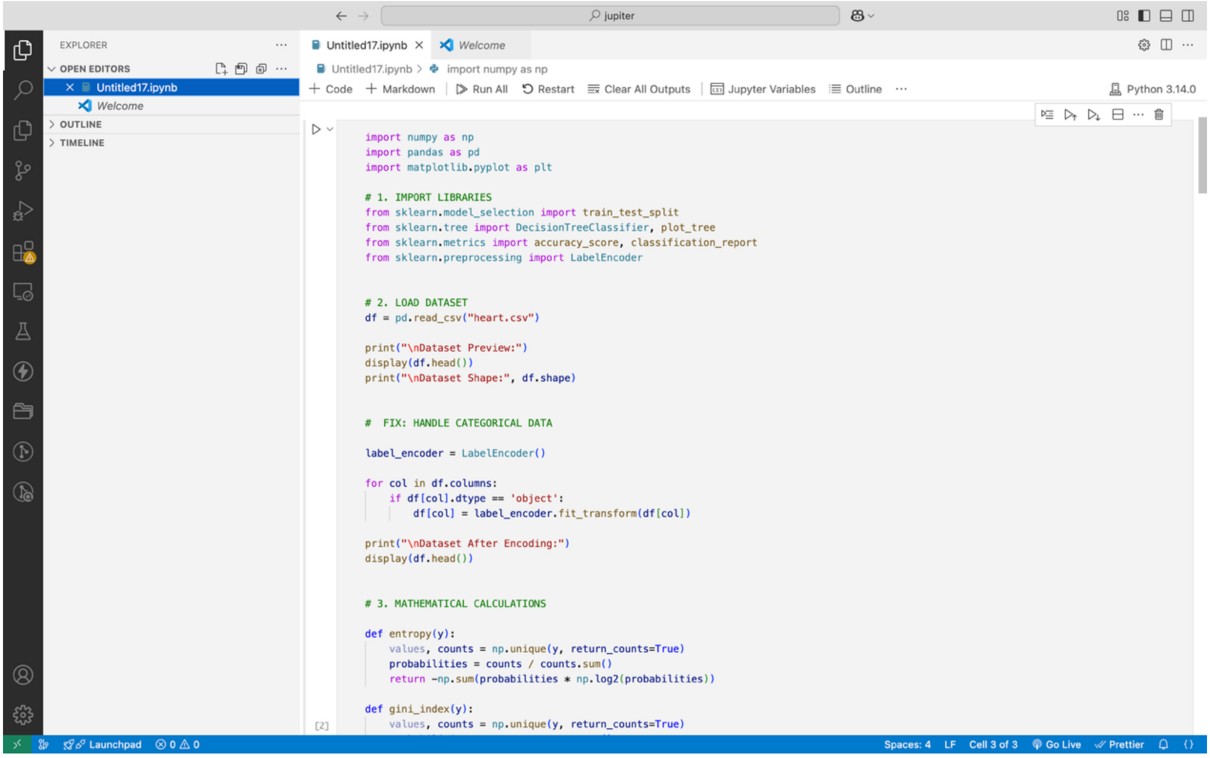
```
label_encoder = LabelEncoder()

for col in df.columns:
    if df[col].dtype == 'object':
        df[col] = label_encoder.fit_transform(df[col])

print("\nDataset After Encoding:")
display(df.head())

# 3. MATHEMATICAL CALCULATIONS

def entropy(y):
    values, counts = np.unique(y, return_counts=True)
    probabilities = counts / counts.sum()
    return -np.sum(probabilities * np.log2(probabilities))

def gini_index(y):
    values, counts = np.unique(y, return_counts=True)
    probabilities = counts / counts.sum()
    return 1 - np.sum(probabilities ** 2)

print("\nEntropy of Target:", entropy(df['target']))
print("Gini Index of Target:", gini_index(df['target']))

# 4. FEATURE & TARGET SPLIT
X = df.drop('target', axis=1)
y = df['target']
```

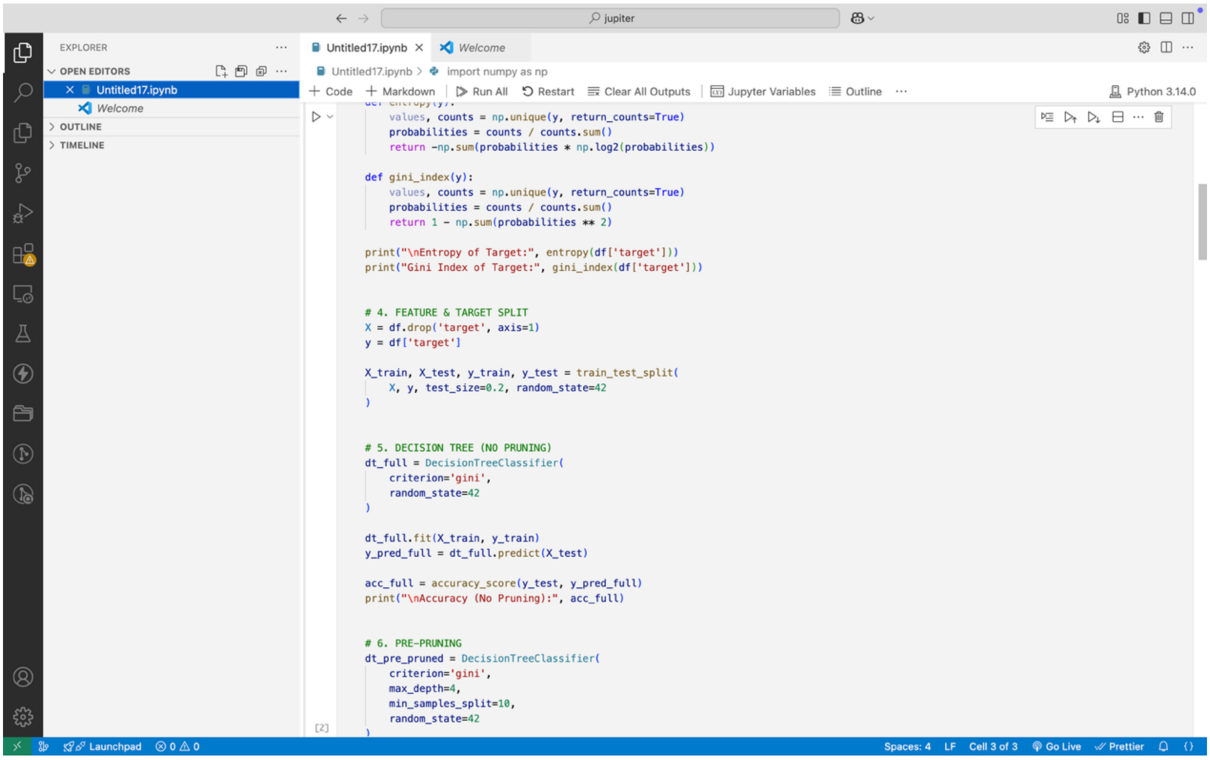
```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# 5. DECISION TREE (NO PRUNING)
dt_full = DecisionTreeClassifier(
    criterion='gini',
    random_state=42
)

dt_full.fit(X_train, y_train)
y_pred_full = dt_full.predict(X_test)

acc_full = accuracy_score(y_test, y_pred_full)
print("\nAccuracy (No Pruning):", acc_full)

# 6. PRE-PRUNING
dt_pre_pruned = DecisionTreeClassifier(
    criterion='gini',
    max_depth=4,
    min_samples_split=10,
    random_state=42
)

dt_pre_pruned.fit(X_train, y_train)
y_pred_pre = dt_pre_pruned.predict(X_test)

acc_pre = accuracy_score(y_test, y_pred_pre)
print("Accuracy (Pre-Pruning):", acc_pre)
```

```
# 7. POST-PRUNING
path = dt_full.cost_complexity_pruning_path(X_train, y_train)
ccp_alphas = path.ccp_alphas

dt_post_pruned = DecisionTreeClassifier(
    random_state=42,
    ccp_alpha=ccp_alphas[5]
)

dt_post_pruned.fit(X_train, y_train)
y_pred_post = dt_post_pruned.predict(X_test)

acc_post = accuracy_score(y_test, y_pred_post)
print("Accuracy (Post-Pruning):", acc_post)

# 8. CLASSIFICATION REPORT
print("\nClassification Report (Pre-Pruned Model):")
print(classification_report(y_test, y_pred_pre))

# 9. DECISION TREE VISUALIZATION
plt.figure(figsize=(20, 10))
plot_tree(
    dt_pre_pruned,
    feature_names=X.columns,
    class_names=['No Disease', 'Disease'],
    filled=True
)
plt.title("Decision Tree Visualization (Pre-Pruned)")
```

```
plt.show()

# 10. FEATURE IMPORTANCE
feature_importance = pd.DataFrame({
    'Feature': X.columns,
    'Importance': dt_pre_pruned.feature_importances_
}).sort_values(by='Importance', ascending=False)

print("\nFeature Importance:")
display(feature_importance)

# 11. FEATURE IMPORTANCE VISUALIZATION
plt.figure(figsize=(10, 5))
plt.bar(feature_importance['Feature'], feature_importance['Importance'])
plt.xticks(rotation=90)
plt.title("Feature Importance Based on Decision Tree")
plt.tight_layout()
plt.show()

# 12. ACCURACY COMPARISON
models = ['No Pruning', 'Pre-Pruning', 'Post-Pruning']
accuracies = [acc_full, acc_pre, acc_post]

plt.figure(figsize=(6, 4))
plt.bar(models, accuracies)
plt.ylabel("Accuracy")
plt.title("Decision Tree Model Accuracy Comparison")
plt.show()
print("\nPROJECT EXECUTED SUCCESSFULLY")
```

Screenshots:



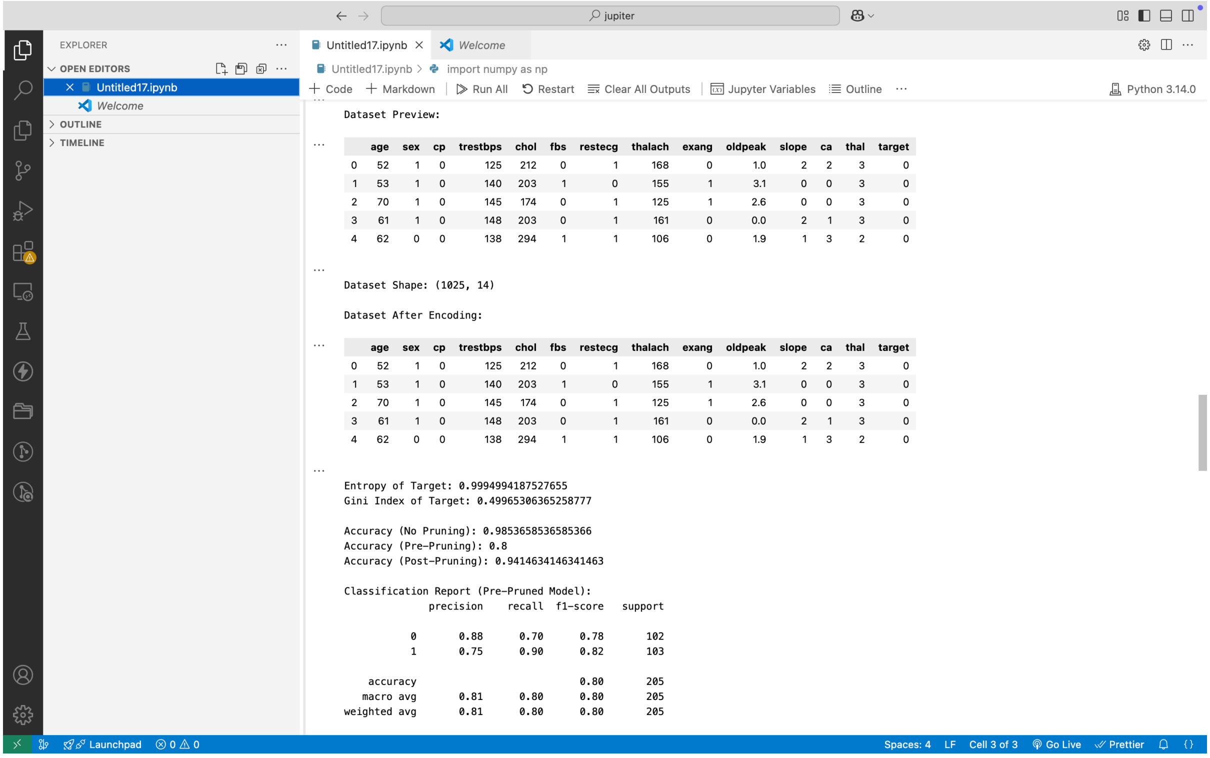


```python
def entropy(y):
    values, counts = np.unique(y, return_counts=True)
    probabilities = counts / counts.sum()
    return -np.sum(probabilities * np.log2(probabilities))

def gini_index(y):
    values, counts = np.unique(y, return_counts=True)
    probabilities = counts / counts.sum()
    return 1 - np.sum(probabilities ** 2)

print("\nEntropy of Target:", entropy(df['target']))
print("Gini Index of Target:", gini_index(df['target']))


# 4. FEATURE & TARGET SPLIT
X = df.drop('target', axis=1)
y = df['target']

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


# 5. DECISION TREE (NO PRUNING)
dt_full = DecisionTreeClassifier(
    criterion='gini',
    random_state=42
)

dt_full.fit(X_train, y_train)
y_pred_full = dt_full.predict(X_test)

acc_full = accuracy_score(y_test, y_pred_full)
print("\nAccuracy (No Pruning):", acc_full)


# 6. PRE-PRUNING
dt_pre_pruned = DecisionTreeClassifier(
    criterion='gini',
    max_depth=4,
    min_samples_split=10,
    random_state=42
)
```

```python
)
plt.title("Decision Tree Visualization (Pre-Pruned)")
plt.show()


# 10. FEATURE IMPORTANCE
feature_importance = pd.DataFrame({
    'Feature': X.columns,
    'Importance': dt_pre_pruned.feature_importances_
}).sort_values(by='Importance', ascending=False)

print("\nFeature Importance:")
display(feature_importance)


# 11. FEATURE IMPORTANCE VISUALIZATION
plt.figure(figsize=(10, 5))
plt.bar(feature_importance['Feature'], feature_importance['Importance'])
plt.xticks(rotation=90)
plt.title("Feature Importance Based on Decision Tree")
plt.tight_layout()
plt.show()


# 12. ACCURACY COMPARISON
models = ['No Pruning', 'Pre-Pruning', 'Post-Pruning']
accuracies = [acc_full, acc_pre, acc_post]

plt.figure(figsize=(6, 4))
plt.bar(models, accuracies)
plt.ylabel("Accuracy")
plt.title("Decision Tree Model Accuracy Comparison")
plt.show()


print("\nPROJECT EXECUTED SUCCESSFULLY")
```

[2] ✓ 0.4s Python

```
Dataset Preview:

age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  slope  ca  thal  target
```



Dataset Preview:

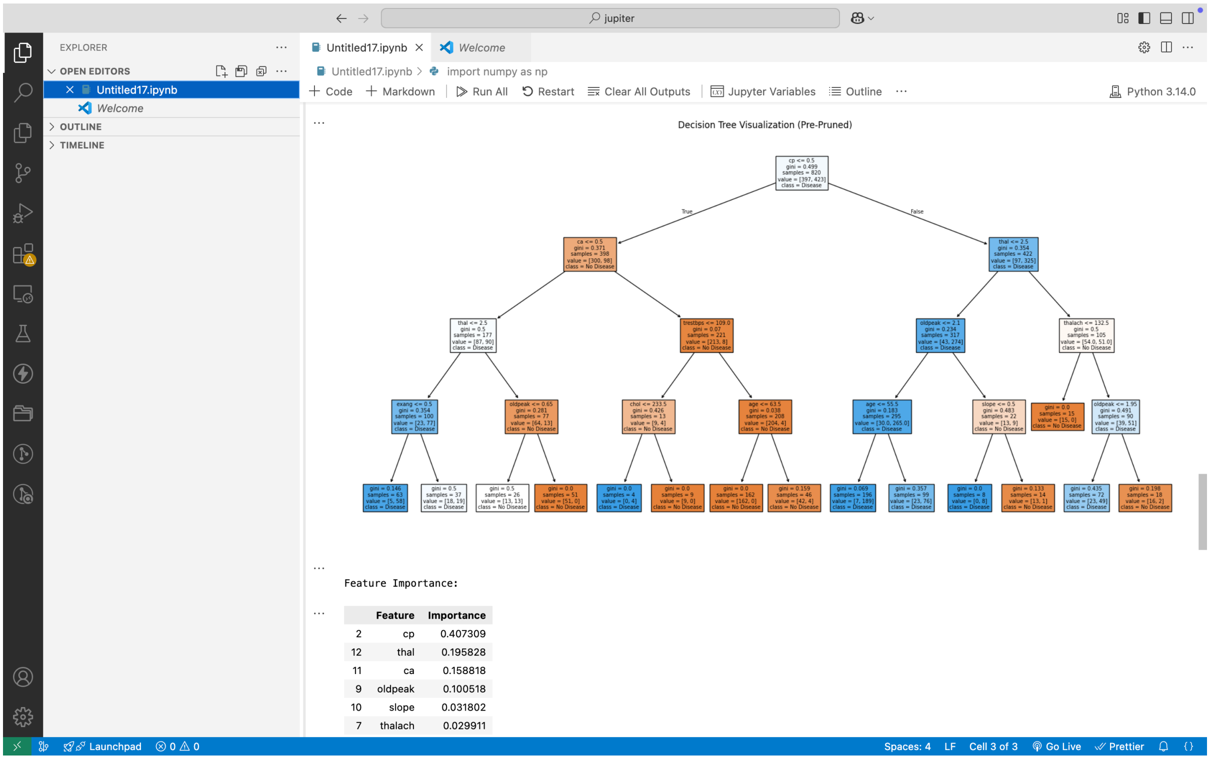
	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0

Dataset Shape: (1025, 14)

Dataset After Encoding:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0

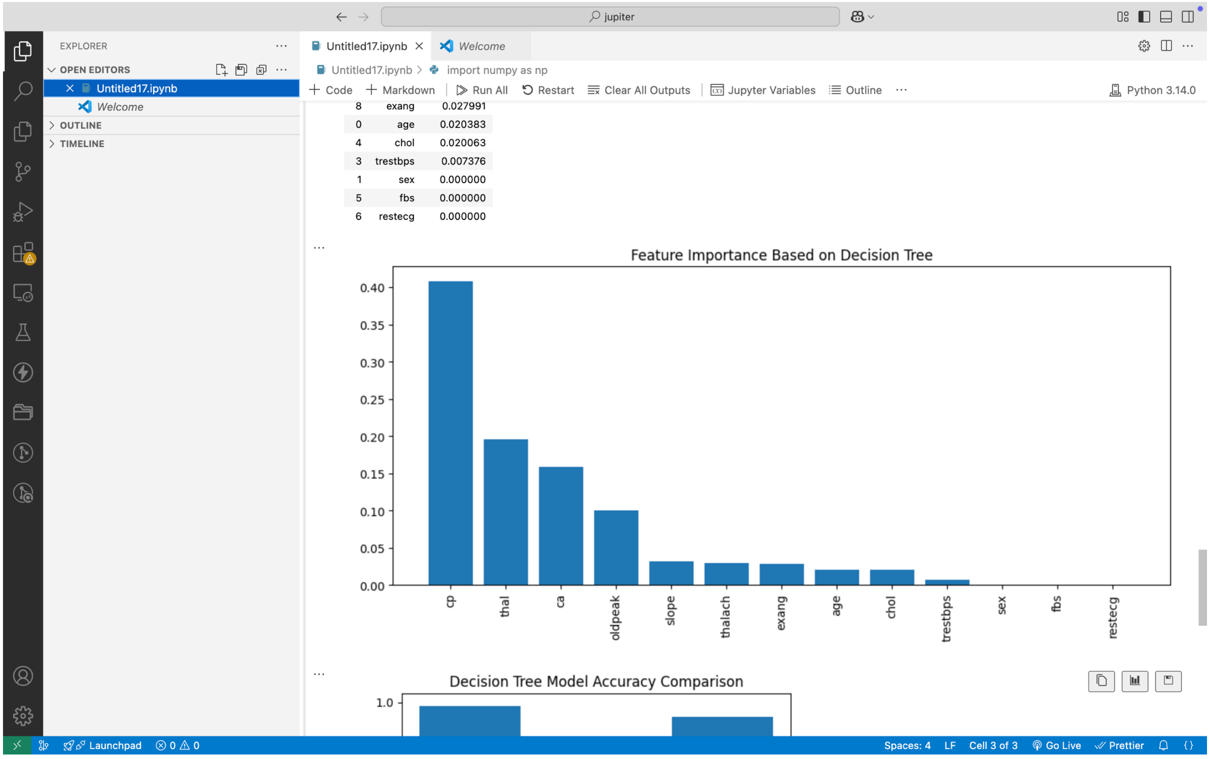
```
Entropy of Target: 0.9994994187527655
Gini Index of Target: 0.49965306365258777

Accuracy (No Pruning): 0.9853658536585366
Accuracy (Pre-Pruning): 0.8
Accuracy (Post-Pruning): 0.9414634146341463

Classification Report (Pre-Pruned Model):
              precision    recall  f1-score   support

           0       0.88      0.70      0.78       102
           1       0.75      0.90      0.82       103

    accuracy                           0.80       205
   macro avg       0.81      0.80      0.80       205
weighted avg       0.81      0.80      0.80       205
```



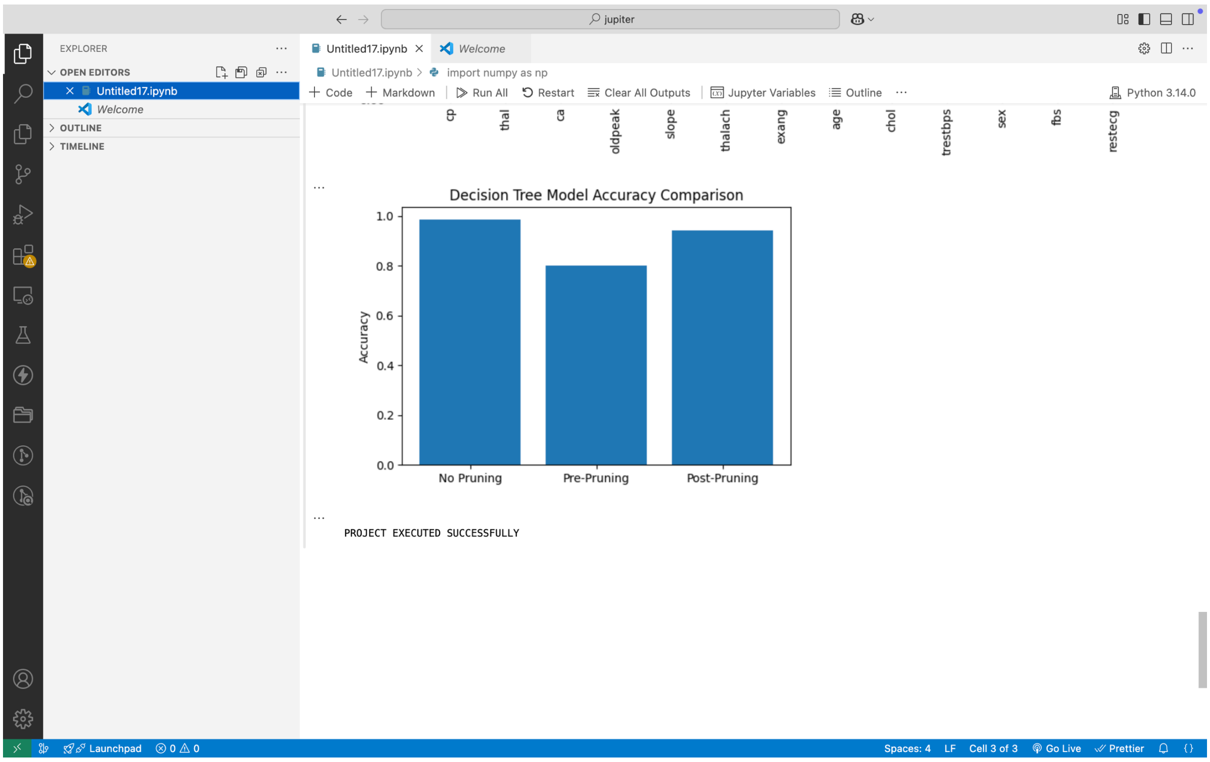
Decision Tree Visualization (Pre-Pruned)

Feature Importance:

	Feature	Importance
2	cp	0.407309
12	thal	0.195828
11	ca	0.158818
9	oldpeak	0.100518
10	slope	0.031802
7	thalach	0.029911



jupiter
EXPLORER
OPEN EDITORS
Untitled17.ipynb
Welcome
OUTLINE
TIMELINE
Untitled17.ipynb
Welcome
Untitled17.ipynb > import numpy as np
+ Code + Markdown Run All Restart Clear All Outputs Jupyter Variables Outline
Python 3.14.0
8 exang 0.027991
0 age 0.020383
4 chol 0.020063
3 trestbps 0.007376
1 sex 0.000000
5 fbs 0.000000
6 restecg 0.000000
Feature Importance Based on Decision Tree
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
cp
thal
ca
oldpeak
slope
thalach
exang
age
chol
trestbps
sex
fbs
restecg
Decision Tree Model Accuracy Comparison
1.0
Launchpad
Spaces: 4 LF Cell 3 of 3 Go Live Prettier



jupiter
EXPLORER
OPEN EDITORS
Untitled17.ipynb
Welcome
OUTLINE
TIMELINE
Untitled17.ipynb
Welcome
Untitled17.ipynb > import numpy as np
+ Code + Markdown Run All Restart Clear All Outputs Jupyter Variables Outline
Python 3.14.0
cp
thal
ca
oldpeak
slope
thalach
exang
age
chol
trestbps
sex
fbs
restecg
Decision Tree Model Accuracy Comparison
1.0
0.8
0.6
0.4
0.2
0.0
Accuracy
No Pruning
Pre-Pruning
Post-Pruning
PROJECT EXECUTED SUCCESSFULLY
Launchpad
Spaces: 4 LF Cell 3 of 3 Go Live Prettier