

Assignment Code: DA-AG-012

Decision Tree | Assignment

Question 1: What is a Decision Tree, and how does it work in the context of classification?

Answer: Logistic Regression is a supervised learning algorithm used for classification tasks (binary or multiclass). It models the probability that a given input belongs to a particular class by applying the logistic (sigmoid) function to a linear combination of input features. The output is a probability in (0,1) , which can be threshold (commonly at 0.5) to produce class predictions.

Difference from linear Regression:

- Task: Linear Regression predicts a continuous numeric value; Logistic regression predicts class probabilities / discrete classes.
- Output Range: Linear regression outputs unbounded real values. Logistic regression outputs values between 0 and 1 (probabilities) after applying the sigmoid.
- Loss / Objective: Linear regression commonly uses mean squared error ; Logistic regression uses log-loss (cross-entropy) derived from likelihood maximization.
- Interpretation: Coefficients in linear regression describe direct linear change in the target. In logistic regression, coefficients relate to log-odds: a one-unit change in a feature changes the log-odds by the coefficient amount.
- Assumptions & Use: Logistic regression assumes a linear relationship between features and the **log-odds** of the outcome, not the outcome itself.

Question 2: Explain the concepts of Gini Impurity and Entropy as impurity measures.

How do they impact the splits in a Decision Tree?

Answer: The Sigmoid function (also called logistic function) is defined as:

$$\sigma(z)=1/(1+e^{-z})$$

where z is the linear combination of inputs ($z=w^T x+b$). The sigmoid transforms any real-valued number into the interval (0, 1), which is interpreted as the probability of the positive class. Without this transformation, the linear model's output would not be a proper probability. The sigmoid also gives the model a convenient differentiable form so we can use gradient-based optimization to minimize cross-entropy loss.

Question 3: What is the difference between Pre-Pruning and Post-Pruning in Decision

Trees? Give one practical advantage of using each.

Answer: Regularization is a technique to prevent overfitting by penalizing large weights in the model. Common forms used in logistic regression:

- **L2 regularization (Ridge):** Adds a penalty proportional to the squared magnitude of coefficients ($\lambda \sum w_i^2$). Encourages small weights but not exactly zero.
- **L1 regularization (Lasso):** Adds a penalty proportional to the absolute value of coefficients ($\lambda \sum |w_i|$). Encourages sparsity and can perform feature selection by driving some weights to zero.

Why needed: Regularization reduces overfitting, improves generalization on unseen data, handles multicollinearity among features, and can control model complexity especially when features > samples or when noisy features exist.

Question 4: What is Information Gain in Decision Trees, and why is it important for choosing the best split?

Answer: Common metrics:

- **Accuracy:** Fraction of correct predictions. Useful when classes are balanced.
- **Precision:** $TP / (TP + FP)$. Of predicted positives, how many are correct. Important when false positives are costly.
- **Recall (Sensitivity):** $TP / (TP + FN)$. Of actual positives, how many did we detect. Important when false negatives are costly.
- **F1-score:** Harmonic mean of precision and recall. Good single metric when there is class imbalance.
- **ROC-AUC:** Area under the ROC curve; measures model's ability to rank positive instances higher than negatives, independent of threshold.
- **PR-AUC:** Area under the precision-recall curve; more informative than ROC-AUC on imbalanced datasets.
- **Confusion Matrix:** Shows TP, TN, FP, FN counts; very helpful for understanding error types.

Why important: Choosing the correct metric depends on business costs associated with false positives vs false negatives and class balance. Using only accuracy on imbalanced data can be misleading.

Question 5: What are some common real-world applications of Decision Trees, and what are their main advantages and limitations?

Answer: # Load dataset, split, train, print accuracy


```
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

using sklearn's breast cancer dataset as CSV substitute

```

data = load_breast_cancer(as_frame=True)
X = data.frame.drop(columns=['target'])
y = data.frame['target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)
model = LogisticRegression(max_iter=1000, solver='liblinear')
model.fit(X_train, y_train)
print('Accuracy:', accuracy_score(y_test, model.predict(X_test)))

```

 Accuracy: 0.956140350877193

Dataset Info:

Question 6: Write a Python program to:

- Load the Iris Dataset
- Train a Decision Tree Classifier using the Gini criterion
- Print the model's accuracy and feature importances

Answer: # Train logistic regression with L2 regularization

```

from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

```

```

model_l2 = LogisticRegression(penalty='l2', C=1.0, solver='liblinear', max_iter=1000)
model_l2.fit(X_train, y_train)
print('First 10 coefficients:', model_l2.coef_[0][:10])
print('Accuracy:', accuracy_score(y_test, model_l2.predict(X_test)))

```

```

First 10 coefficients: [ 1.93035716e+00  7.14564675e-02 -5.03933094e-02 -1.6957
-1.45563995e-01 -3.81326446e-01 -5.93761860e-01 -3.03860472e-01
-2.66197475e-01 -2.89831558e-02]
Accuracy: 0.956140350877193

```

Question 7: Write a Python program to:

- Load the Iris Dataset
- Train a Decision Tree Classifier with max_depth=3 and compare its accuracy to a fully-grown tree.

Answer: # Multiclass classification with multi_class='ovr' on iris

```
from sklearn.datasets import load_iris

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import classification_report

from sklearn.model_selection import train_test_split


iris = load_iris(as_frame=True)

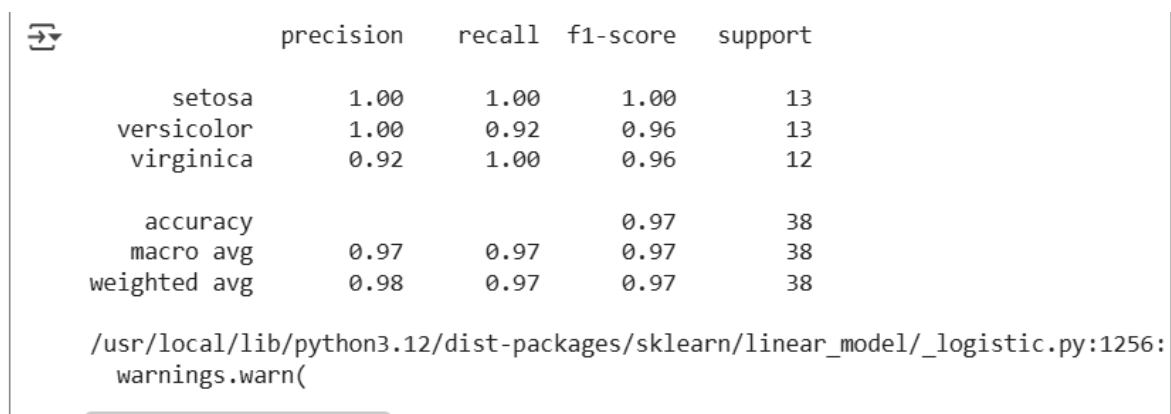
X = iris.data; y = iris.target

Xtr, Xte, ytr, yte = train_test_split(X, y, test_size=0.25, random_state=0, stratify=y)

model_ovr = LogisticRegression(multi_class='ovr', solver='liblinear', max_iter=1000)

model_ovr.fit(Xtr, ytr)

print(classification_report(yte, model_ovr.predict(Xte), target_names=iris.target_names))
```



	precision	recall	f1-score	support
setosa	1.00	1.00	1.00	13
versicolor	1.00	0.92	0.96	13
virginica	0.92	1.00	0.96	12
accuracy			0.97	38
macro avg	0.97	0.97	0.97	38
weighted avg	0.98	0.97	0.97	38

/usr/local/lib/python3.12/dist-packages/sklearn/linear_model/_logistic.py:1256:
warnings.warn(

Question 8: Write a Python program to:

- Load the California Housing dataset from sklearn
- Train a Decision Tree Regressor
- Print the Mean Squared Error (MSE) and feature importances

Answer: # GridSearchCV tuning C and penalty

```
from sklearn.model_selection import GridSearchCV

from sklearn.linear_model import LogisticRegression


param_grid = {'C': [0.01, 0.1, 1, 10], 'penalty': ['l1', 'l2']}

grid = GridSearchCV(LogisticRegression(solver='liblinear', max_iter=1000), param_grid, cv=5,
                    scoring='accuracy')

grid.fit(X_train, y_train)
```

```
print('Best params:', grid.best_params_)  
print('Best CV accuracy:', grid.best_score_)
```

```
⇒ Best params: {'C': 10, 'penalty': 'l1'}  
Best CV accuracy: 0.9582417582417582
```

Question 9: Write a Python program to:

- Load the Iris Dataset
- Tune the Decision Tree's max_depth and min_samples_split using GridSearchCV
- Print the best parameters and the resulting model accuracy

Answer: # Compare with and without scaling

```
from sklearn.preprocessing import StandardScaler  
from sklearn.linear_model import LogisticRegression  
from sklearn.metrics import accuracy_score  
  
scaler = StandardScaler()  
X_train_s = scaler.fit_transform(X_train)  
X_test_s = scaler.transform(X_test)  
model_no_scale = LogisticRegression(solver='liblinear', max_iter=1000).fit(X_train, y_train)  
model_scaled = LogisticRegression(solver='liblinear', max_iter=1000).fit(X_train_s, y_train)  
print('Accuracy without scaling:', accuracy_score(y_test, model_no_scale.predict(X_test)))  
print('Accuracy with scaling:', accuracy_score(y_test, model_scaled.predict(X_test_s)))
```

```
print('Accuracy with scaling:', accuracy_score(y_test, model_scaled.predict(X_test_s)))  
⇒ Accuracy without scaling: 0.956140350877193  
Accuracy with scaling: 0.9824561403508771
```

Question 10: Imagine you're working as a data scientist for a healthcare company that wants to predict whether a patient has a certain disease. You have a large dataset with mixed data types and some missing values.

Explain the step-by-step process you would follow to:

- Handle the missing values

- Encode the categorical features
- Train a Decision Tree model
- Tune its hyperparameters
- Evaluate its performance

And describe what business value this model could provide in the real-world setting.

Answer:

When the positive class (responders) is only 5%, special care is required. Here's a step-by-step approach:

1. Understand business objective & costs

- Clarify the relative cost of false positives (wasting marketing spend) vs false negatives (missed revenue). This guides metric selection and thresholding.

2. Data handling & exploration

- Explore feature distributions, missing values, outliers, and correlations.
- Create domain-derived features (recency/frequency/monetary (RFM) for e-commerce), interaction terms, and temporal features.

3. Feature preprocessing and scaling

- Impute missing values appropriately (median for skewed numeric, mode for categorical, or model-based imputation).
- Encode categorical variables (one-hot for low cardinality, target encoding or ordinal encoding for high-cardinality categories, with cross-validation to avoid leakage).
- Standardize/normalize numeric features if using regularization or gradient-based solvers.

4. Address class imbalance Options (often combined for best effect):

- **Resampling:**
 - *Oversampling* (e.g., SMOTE variants)