Evaluating and Optimizing Decision Trees: Takeaways

Syntax

1) EVALUATION OF DECISION TREES

a) Regression Metrics

- R² Score — First Method

```
reg_tree = DecisionTreeRegressor()
reg_tree.fit(X_train, y_train)
reg_tree.score(X_test, y_test)
```

- R² Score — Second Method

```
from sklearn.metrics import r2_score
reg_tree = DecisionTreeRegressor()
reg_tree.fit(X_train, y_train)
y_pred = reg_tree.predict(X_test)
r2_score(y_test, y_pred)
```

- RMSE Using `mean_squared_error`

```
from sklearn.metrics import mean_squared_error
mean_squared_error(y_test, y_pred, squared = False)
```

b) Classification Metrics

- Accuracy Score — First Method

```
class_tree = DecisionTreeClassifier()
class_tree.fit(X_train, y_train)
class_tree.score(X_test, y_test)
```

- Accuracy Score — Second Method

```
from sklearn.metrics import accuracy_score
class_tree = DecisionTreeClassifier()
class_tree.fit(X_train, y_train)
y_pred = class_tree.predict(X_test)
accuracy_score(y_test, y_pred)
```

- Multilabel Confusion Matrix

```
from sklearn.metrics import multilabel_confusion_matrix
multilabel_confusion_matrix(y_test, y_pred, labels=["Class 1", "Class 2", "Class 3"])
```

```
array([[[TN - True Negative, FP - False Positive], [FN - False Negative, TP - True Positive]],
# Class 1 [[TN - True Negative, FP - False Positive], [FN - False Negative, TP - True
```

`Positive]], # Class 2 [[TN - True Negative, FP - False Positive], [FN - False Negative, TP - True Positive]]]) # Class 3`

- Classification Report

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

2) OPTIMIZATION OF DECISION TREES

- Plotting Feature Importances

```
import matplotlib.pyplot as plt
plt.rcParams["figure.figsize"] = [11.0, 10.0]
plt.barh(decision_tree.feature_names_in_, decision_tree.feature_importances_))
plt.xlabel("Importance (0 - 1)")
plt.ylabel("Column")
plt.title("Feature Importance for Decision Tree")
plt.show()
```

- Applying Minimal Cost-Complexity Pruning to an unpruned Tree

```
from sklearn.tree import < Decision Tree >
import matplotlib.pyplot as plt
cost_tree = DecisionTree(random_state = 14)
cost_path = cost_tree.cost_complexity_pruning_path(X_train, y_train)
cost_path.ccp_alphas
subtrees = []
for alpha in cost_path.ccp_alphas:
    subtree = DecisionTree(random_state = 14, ccp_alpha = alpha)
    subtree.fit(X_train, y_train)
    subtrees.append(subtree)
ccp_alphas_slice = cost_path.ccp_alphas[-10:-3]
subtrees_slice = subtrees[-10:-3] # Both slices are required to match.
subtree_scores = [subtree.score(X_test, y_test) for subtree in subtrees_slice]
alpha_scores = tuple(zip(ccp_alphas_slice, subtree_scores))
print(max(alpha_scores, key=lambda x:x[1]))
```

- Minimal Cost Complexity Pruning - Plotting the subtree scores per alpha value

```
plt.rcParams["figure.figsize"] = [10.0, 6.0]
fig, ax = plt.subplots()
ax.set_xlabel("Alpha")
ax.set_ylabel("Score")
ax.set_title("Score per Alpha Value")
ax.plot(ccp_alphas_slice, subtree_scores, marker="o")
plt.show()
```

Concepts

1) EVALUATION OF DECISION TREES

a) Regression Metrics

RMSE

- The RMSE metric is simply the square root of the mean squared error (MSE). It is calculated by finding the sum of square differences between the actual values and the predicted values, and then dividing by the total number of observations before taking the square root of the result.
  $\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\mathrm{y\_test}_i - \mathrm{y\_pred}_i)^2}$
  where n represents the total number of observations and the $\sum$ expression calculates the total sum of squared differences between the actual values (`y_test`) and the predicted values (`y_pred`). When we divide this sum by n, we get the MSE. Lastly, taking the square root of the MSE gives us the RMSE.

R^2

- R^2, also known as the **coefficient of determination**, measures how far away the predictions of the regression tree are from the real values, just like the RMSE. The advantage of R^2 is that it doesn't depend on the scale of the data. It's the default evaluation metric used for regression trees. R^2 is a percentage that represents the proportion of the variance we see in our predictions that can be explained by the model.
  $R^2 = 1 - \frac{RSS}{TSS}$
  Where:
  $RSS = \sum_{i=1}^{n}\left(\mathrm{y\_test}_i - \mathrm{y\_pred}_i\right)^2$
  $TSS = \sum_{i=1}^{n}\left(\mathrm{y\_test}_i - \overline{\mathrm{y\_test}}\right)^2$

---

b) Classification Metrics

- A Multilabel Confusion Matrix generates a matrix for each class of the target column.
  - **True positives** are the observations where the class is correctly predicted; in other words the class appears in both `y_test` and `y_pred` .
  - **True negatives** are the observations where the class is correctly not predicted; in other words the class doesn't appear in either `y_test` or `y_pred` .

    - **False positives/Type I Errors** are observations where the class is actually absent, but the model wrongly predicts that it's present; in other words, the class will appear in `y_pred` but not in `y_test`.
    - **False negatives/Type II Errors** are observations where the class is actually present, but the model wrongly predicts that it's absent; in other words, the class won't appear in `y_pred`, but it will be in `y_test`.
- **Accuracy**: the default evaluation metric for classification trees. It is the total number of correct predictions (where `y_test` and `y_pred` match) divided by the total number of predictions.

$$\text{Accuracy} = \frac{(TP + TN)}{(TP + TN + FP + FN)}$$

- In the case of binary target columns we also have these metrics:
    - **Precision**: the ratio of correctly classified positive predictions to the total number of predicted positives.

$$\text{Precision} = \frac{TP}{(TP + FP)}$$

    - **Recall**: the ratio of correctly classified positive predictions to the total number of actual positives.

$$\text{Recall} = \frac{TP}{(TP + FN)}$$

    - **F1-Score**: the harmonic mean of the previous two metrics.

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

- The classification report also provides two additional metrics:
    - The `macro avg` (macro average) is obtained by summing all of the metrics for a specific column and then dividing by the number of distinct classes.
    - The `weighted avg` (weighted average) is obtained by taking the value in `support` (i.e., the number of observations in `y_test` with that specific class) and dividing it by the value that repeats three times at the bottom of the `support` column (which represents the total number of observations) to give you the **weight**, which is then multiplied by the selected metric in that specific class. After this, we repeat these previous steps for the rest of the classes, and once we have all the results for each class, we sum them all.

---

2) OPTIMIZATION OF DECISION TREES

- There are two main categories of pruning:
    - **Prepruning**: establishing conditions in which the decision tree stops generating thresholds, before finishing the recursive process. One example of this is the parameter `max_depth`.
    - **Postpruning**: as the name implies, this category encompasses all the optimization techniques applied after the decision tree was built.
- Both regression and classification trees share the same optimization parameters:
    - `max_depth`: limits the number of levels in the decision tree. `3` is the default value. Usually the first optimization parameter to set.
    - `criterion`:
        - **Regression Trees**: `squared_error` (*Mean Squared Error - MSE*), `absolute_error` (*Mean Absolute Error - MAE*), `friedman_mse`, `poisson`.

        - **Classification Trees**: `gini` , `entropy` , `log_loss` .
    - `min_samples_split` : minimal number of observations required to split an internal node.
    - `min_samples_leaf` : minimal number of observations required to be present at a leaf node.
    - `max_features` : number of feature columns to consider when searching for the best split.
    - `max_leaf_nodes` : establishes the maximum number of leaves the tree can have. The nodes created will be decided based on their relative reduction in impurity — in other words, the best ones will have priority.
    - `min_impurity_decrease` : a node will only be split if it causes a reduction in the weighted impurity that is greater than or equal to the value set in this parameter.
    - `ccp_alpha` : value used during Minimal Cost Complexity Pruning, which sets a penalty to decision trees with a high number of leaves in order to reduce overfitting.
        - Minimal Cost-Complexity Pruning is an optimization technique that generates a series of subtrees with different number of leaves, and then for each subtree it sums the impurity or misclassification rate of all the leaves, and finally it sums to that calculation the number of leaves multiplied by the value of the `ccp_alpha` parameter, which is called **alpha**. This kind of pruning aims to find the subtree with the lowest impurity/misclassification rate. This is how the calculation is performed on every subtree:

$$\text{Subtree}_N = \sum \text{Impurity of All Leaves} + (\text{Number of Leaves} \times \text{ccp\_alpha})$$

    - `class_weight` *(Exclusive to Classification Trees)* : Assigns weights to each class in multilabel target columns to compensate for unbalanced frequency of classes.
    - `feature_importances_` *(Attribute)* : Outputs an array with a normalized value for every column, which indicates how much that column contributes with useful thresholds — and by extension, how much predictive power it offers.

Resources

1) EVALUATION OF DECISION TREES

a) Regression Metrics

- R2 Score - Overview
    - R2 Score
    - Regression Tree's R2 Score
    - Wikipedia: R2 Score / Coefficient of determination
- Mean Squared Error - Overview
    - Mean Squared Error
    - Wikipedia: Mean Squared Error
    - Wikipedia: Root Mean Square Deviation
- Additional metrics:
    - Mean Absolute Error
    - Median Absolute Error

---

b) Classification Metrics

- Accuracy Score
    - Classification Tree's Accuracy Score
    - Accuracy in scikit-learn's documentation
    - Wikipedia: Accuracy and Precision
- Classification Report - Overview
    - Classification Report
- Binary Classification
    - Multiclass and Multilabel Classification
- Additional Metrics - Overview:
    - Precision
    - Recall
    - Wikipedia: Precision and Recall
    - F1 Score
    - F-Score
- Multilabel Confusion Matrix - Overview
    - Multilabel Confusion Matrix

---

2) OPTIMIZATION OF DECISION TREES

- Overfitting
- Bias / Variance Tradeoff
- Regression Tree's Parameters
- Classification Tree's Parameters
- train_test_split Parameters
    - Advanced Stratification Methods
- Feature Importances Evaluation
    - Classification Tree's Feature Importances
    - Regression Tree's Feature Importances (NOTE: It's similar to the previous item)
- Pruning of Decision Trees
- Minimal Cost Complexity Pruning
- Post pruning Decision Trees with Cost Complexity Pruning
- scikit-learn Tips for Decision Trees