Q4.) Understand the source code of DecisionTreeClassifier

a.) Please denote two strategies that this classifier implements to pre-prune or post-prune the tree.

Ans.) Pre-Pruning or Post-Pruning strategies:

1.) class sklearn.tree.DecisionTreeClassifier(max_features : int, float or {"auto", "sqrt", "log2"}), default=None

The number of features to consider when looking for the best split:

- If int, then consider `max_features` features at each split.

- If float, then `max_features` is a fraction and

`int(max_features * n_features)` features are considered at each split.

- If "auto", then `max_features=sqrt(n_features)`.

- If "sqrt", then `max_features=sqrt(n_features)`.

- If "log2", then `max_features=log2(n_features)`.

- If None, then `max_features=n_features`.

2.) class sklearn.tree.DecisionTreeClassifier(min_samples_leaf : int or float), default=1

The minimum number of samples required to be at a leaf node.

A split point at any depth will only be considered if it leaves at least `min_samples_leaf` training samples in each of the left and right branches. This may have the effect of smoothing the model, especially in regression.

- If int, then consider `min_samples_leaf` as the minimum number.

- If float, then `min_samples_leaf` is a fraction and `ceil(min_samples_leaf * n_samples)` are the minimum number of samples for each node.

b.) For each strategy, please clearly identify the repository file and the lines of code that implement such strategies.

Ans.)

1.) For max_features,

```
if isinstance(self.max_features, str):
        if self.max_features == "auto":
            if is_classification:
                max_features = max(1, int(np.sqrt(self.n_features_in_)))
            else:
                max_features = self.n_features_in_
        elif self.max_features == "sqrt":
            max_features = max(1, int(np.sqrt(self.n_features_in_)))
```

```
            elif self.max_features == "log2":
                max_features = max(1, int(np.log2(self.n_features_in_)))
            else:
                raise ValueError(
                    "Invalid value for max_features. "
                    "Allowed string values are 'auto', "
                    "'sqrt' or 'log2'."
                )
        elif self.max_features is None:
            max_features = self.n_features_in_
        elif isinstance(self.max_features, numbers.Integral):
            max_features = self.max_features
        else:  # float
            if self.max_features > 0.0:
                max_features = max(1, int(self.max_features * self.n_features_in_))
            else:
                max_features = 0

        self.max_features_ = max_features
```

2.) For min_samples_leaf,

```
    if isinstance(self.min_samples_leaf, numbers.Integral):
            if not 1 <= self.min_samples_leaf:
                raise ValueError(
                    "min_samples_leaf must be at least 1 or in (0, 0.5], got %s"
                    % self.min_samples_leaf
                )
            min_samples_leaf = self.min_samples_leaf
        else:  # float
            if not 0.0 < self.min_samples_leaf <= 0.5:
                raise ValueError(
                    "min_samples_leaf must be at least 1 or in (0, 0.5], got %s"
                    % self.min_samples_leaf
                )
            min_samples_leaf = int(ceil(self.min_samples_leaf * n_samples))
```