In [82]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV, KFold
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
```

In [83]:
```
data = pd.read_csv(r"Practical6.csv")
data.head()
```

Out[83]:

	outlook	temp	humidity	windy	play
0	sunny	hot	high	False	no
1	sunny	hot	high	True	no
2	overcast	hot	high	False	yes
3	rainy	mild	high	False	yes
4	rainy	cool	normal	False	yes

In [84]:
```
X = data.iloc[:, [1, 2, 3, 4]].values
y = data.iloc[:, -1].values
```

In [85]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
X[:,0] = le.fit_transform(X[:,0])
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra
```

In [86]:
```
dt_classifier = DecisionTreeClassifier()
```

In [87]:
```
param_grid = {
    'criterion': ['gini', 'entropy'],
    'max_depth': [3, 5, 7, 10],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
```

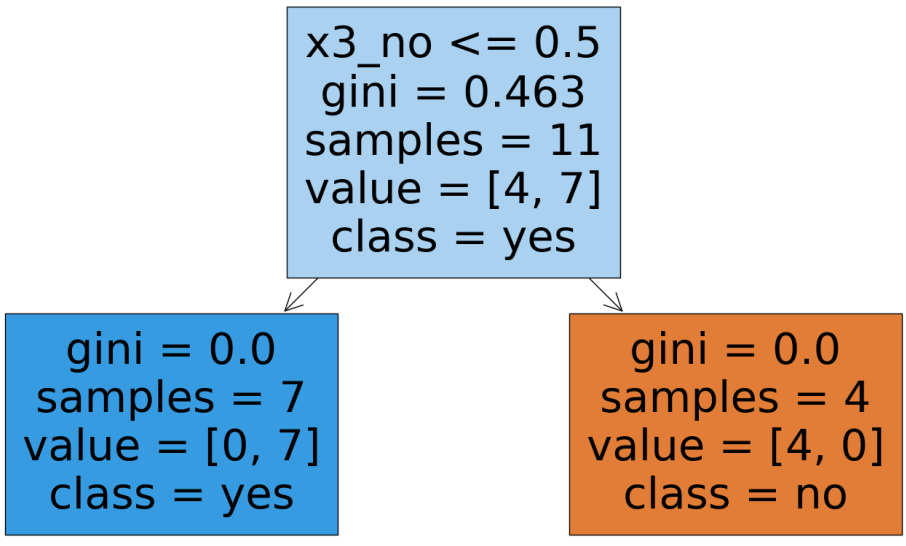
In [88]:
```
kf = KFold(n_splits=5, shuffle=True, random_state=42)
```

In [89]:
```
from sklearn.preprocessing import OneHotEncoder

# Create an instance of the OneHotEncoder
encoder = OneHotEncoder()

# Fit and transform the categorical columns
X_train_encoded = encoder.fit_transform(X_train)
X_test_encoded = encoder.transform(X_test)
```

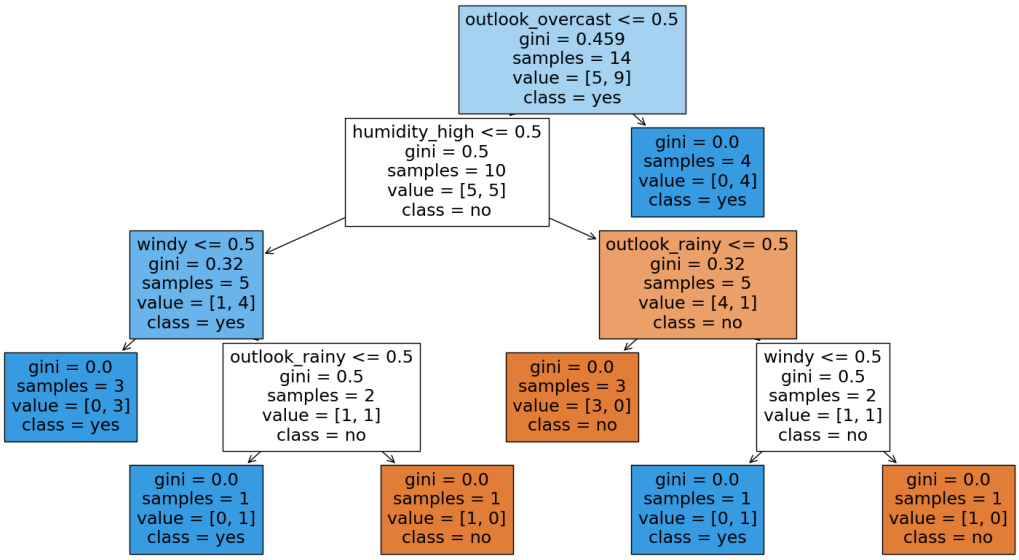
In [90]:
```
# Create a GridSearchCV object
grid_search = GridSearchCV(dt_classifier, param_grid, cv=kf, scoring='accur

# Fit the GridSearchCV object to the training data
grid_search.fit(X_train_encoded, y_train)

# Get the best parameters and best score
best_params = grid_search.best_params_
best_score = grid_search.best_score_

# Instantiate the Decision Tree Classifier with the best parameters
best_dt_classifier = DecisionTreeClassifier(**best_params)

# Train the classifier with the entire training data
best_dt_classifier.fit(X_train_encoded, y_train)

# Make predictions on the test data
y_pred = best_dt_classifier.predict(X_test_encoded)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
print("Best Parameters:", best_params)
print("Best Score:", best_score)
```

```
Accuracy: 1.0
Best Parameters: {'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf':
1, 'min_samples_split': 2}
Best Score: 1.0
```

In [91]:
```
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt

# Plot the decision tree
plt.figure(figsize=(20,10))
plot_tree(best_dt_classifier, feature_names=encoder.get_feature_names_out()
plt.show()
```

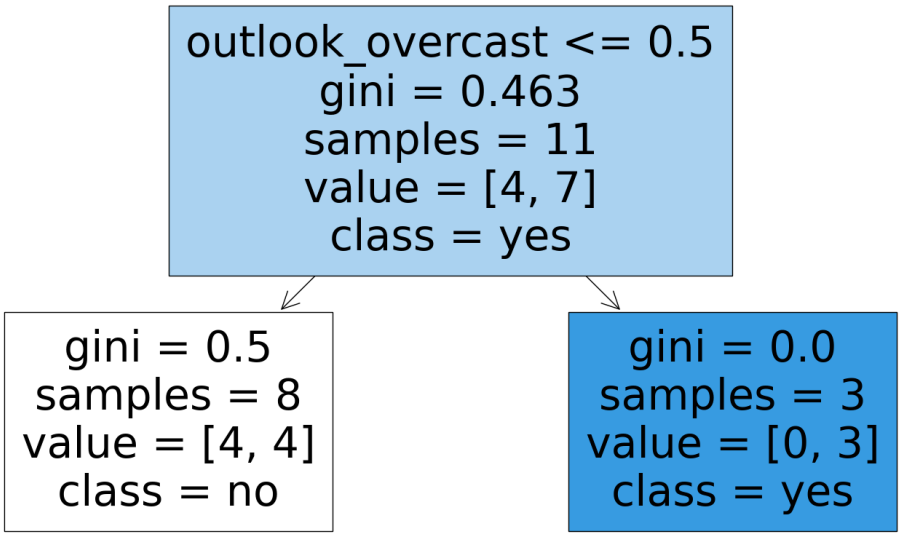


In [92]:
```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier, plot_tree
import matplotlib.pyplot as plt

# Step 1: Load the Data
data = pd.read_csv("practical6.csv")

# Step 2: Preprocess Categorical Variables
# Assuming all features except the target column are categorical
X = data.drop(columns=['play'])  # Features
y = data['play']  # Target

# Convert categorical variables into numerical format using one-hot encodin
X_encoded = pd.get_dummies(X)

# Step 3: Train Decision Tree Classifier
dt_classifier = DecisionTreeClassifier()
dt_classifier.fit(X_encoded, y)

# Step 4: Print Decision Tree
plt.figure(figsize=(20, 10))
plot_tree(dt_classifier, feature_names=X_encoded.columns, class_names=['no'
plt.show()
```



In [93]:
```
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV, KFold
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

# Step 1: Load the Data
data = pd.read_csv("practical6.csv")

# Step 2: Preprocess the Data
X = data.drop(columns=['play'])  # Features
y = data['play']  # Target
X_encoded = pd.get_dummies(X)

# Step 3: Split the Data into Training and Test Sets
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, test_size

# Step 4: Define the Decision Tree Classifier
dt_classifier = DecisionTreeClassifier()

# Step 5: Define the Hyperparameter Grid for GridSearchCV
param_grid = {
    'criterion': ['gini', 'entropy'],
    'max_depth': [3, 5, 7, 10],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# Step 6: Perform GridSearchCV with K Fold Cross-Validation
kf = KFold(n_splits=5, shuffle=True, random_state=42)
grid_search = GridSearchCV(dt_classifier, param_grid, cv=kf, scoring='accur
grid_search.fit(X_train, y_train)

# Step 7: Get the Best Model and Evaluate on Test Set
best_dt_classifier = grid_search.best_estimator_
y_pred = best_dt_classifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)

# Step 8: Print Results
print("Best Parameters:", grid_search.best_params_)
print("Best Score (CV Accuracy):", grid_search.best_score_)
print("Accuracy on Test Set:", accuracy)

import matplotlib.pyplot as plt
from sklearn.tree import plot_tree

# Plot the decision tree of the best parameters
plt.figure(figsize=(20, 10))
plot_tree(best_dt_classifier, feature_names=X_encoded.columns, class_names=
plt.show()
```

```
Best Parameters: {'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf':
1, 'min_samples_split': 10}
Best Score (CV Accuracy): 0.6333333333333333
Accuracy on Test Set: 0.6666666666666666
```


outlook_overcast <= 0.5
gini = 0.463
samples = 11
value = [4, 7]
class = yes
gini = 0.5
samples = 8
value = [4, 4]
class = no
gini = 0.0
samples = 3
value = [0, 3]
class = yes