Practical 6

Aim : To implement a Machine Learning Classification model using a Decision Tree Classifier algorithm and enhance the model by K Fold and GridSearchCV cross-validation.

In [15]:
```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier, plot_tree
import matplotlib.pyplot as plt
```

Step 1: Load the Data

In [16]:
```
data = pd.read_csv("practical6.csv")
X = data.drop(columns=['play'])  # Features
y = data['play']  # Target
```

Convert categorical variables into numerical format using one-hot encoding

In [17]:
```
X_encoded = pd.get_dummies(X)
```

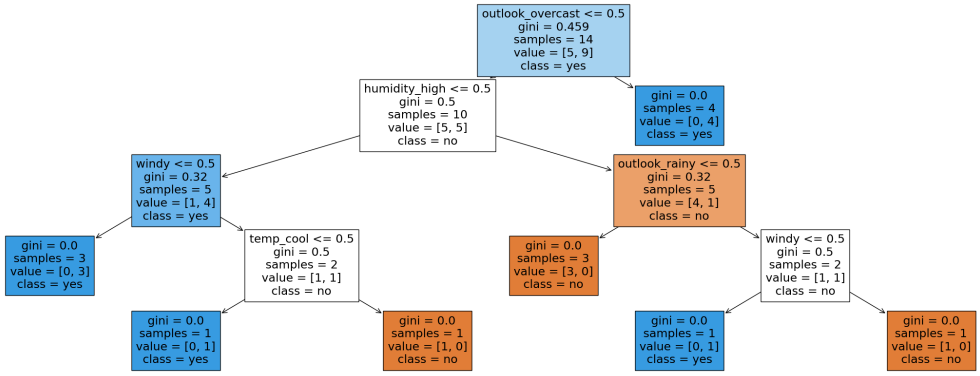
Step 3: Train Decision Tree Classifier

In [18]:
```
dt_classifier = DecisionTreeClassifier()
dt_classifier.fit(X_encoded, y)
```

Out[18]:
```
DecisionTreeClassifier
DecisionTreeClassifier()
```

Step 4: Print Decision Tree

In [19]:
```
plt.figure(figsize=(30, 10))
plot_tree(dt_classifier, feature_names=X_encoded.columns, class_names=['no'
plt.show()
```



GridSearchCV And KFold CrossValidation Implementation

In [20]:
```
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 [21]:
```
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, test_size
```

Step 4: Define the Decision Tree Classifier

In [22]:
```
dt_classifier = DecisionTreeClassifier()
```

Step 5: Define the Hyperparameter Grid for GridSearchCV

In [23]:
```
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

In [24]:
```
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)
```

Out[24]:

```
GridSearchCV(cv=KFold(n_splits=5, random_state=42, shuffle=True),
             estimator=DecisionTreeClassifier(),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [3, 5, 7, 10],
                         'min_samples_leaf': [1, 2, 4],
                         'min_samples_split': [2, 5, 10]},
             scoring='accuracy')
```

estimator: DecisionTreeClassifier

```
DecisionTreeClassifier()
```

DecisionTreeClassifier

```
DecisionTreeClassifier()
```

Step 7: Get the Best Model and Evaluate on Test Set

In [25]:
```
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

In [26]:
```
print("Best Parameters:", grid_search.best_params_)
print("Best Score (CV Accuracy):", grid_search.best_score_)
print("Accuracy on Test Set:", accuracy)
```

```
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
```

Plot the decision tree of the best parameters

In [27]:
```
import matplotlib.pyplot as plt
from sklearn.tree import plot_tree


plt.figure(figsize=(20, 10))
plot_tree(best_dt_classifier, feature_names=X_encoded.columns, class_names=
plt.show()
```