In [98]:
```
import pandas as pd
```

In [99]:
```
df = pd.read_csv("golf_dataset_long_format.csv")
df.head(20)
```

Out[99]:

	Temperature	Humidity	Wind	Outlook	Play
0	3.3	49	1	3	1
1	3.3	49	1	3	0
2	3.3	49	1	3	0
3	3.3	49	1	3	1
4	3.3	49	1	3	1
5	3.3	49	1	3	0
6	3.3	49	1	3	0
7	2.9	53	0	3	1
8	2.9	53	0	3	0
9	2.9	53	0	3	1
10	2.9	53	0	3	0
11	2.9	53	0	3	0
12	2.9	53	0	3	0
13	2.9	53	0	3	0
14	2.3	58	0	1	0
15	2.3	58	0	1	0
16	2.3	58	0	1	1
17	2.3	58	0	1	0
18	2.3	58	0	1	0
19	2.3	58	0	1	0

In [100…
```
cols = list(df.columns)
print(cols)
```

```
['Temperature', 'Humidity', 'Wind', 'Outlook', 'Play']
```

In [101…
```
df.shape
```

Out[101…
```
(7665, 5)
```

In [102…
```
corrmat = df.corr()
top_corr_features = corrmat.index
```

```
corrmat
```

Out[102…

	Temperature	Humidity	Wind	Outlook	Play
Temperature	1.000000	0.683181	-0.162446	-0.113775	-0.021652
Humidity	0.683181	1.000000	-0.115711	-0.139317	-0.096551
Wind	-0.162446	-0.115711	1.000000	-0.028279	-0.054290
Outlook	-0.113775	-0.139317	-0.028279	1.000000	0.068390
Play	-0.021652	-0.096551	-0.054290	0.068390	1.000000

In [103…

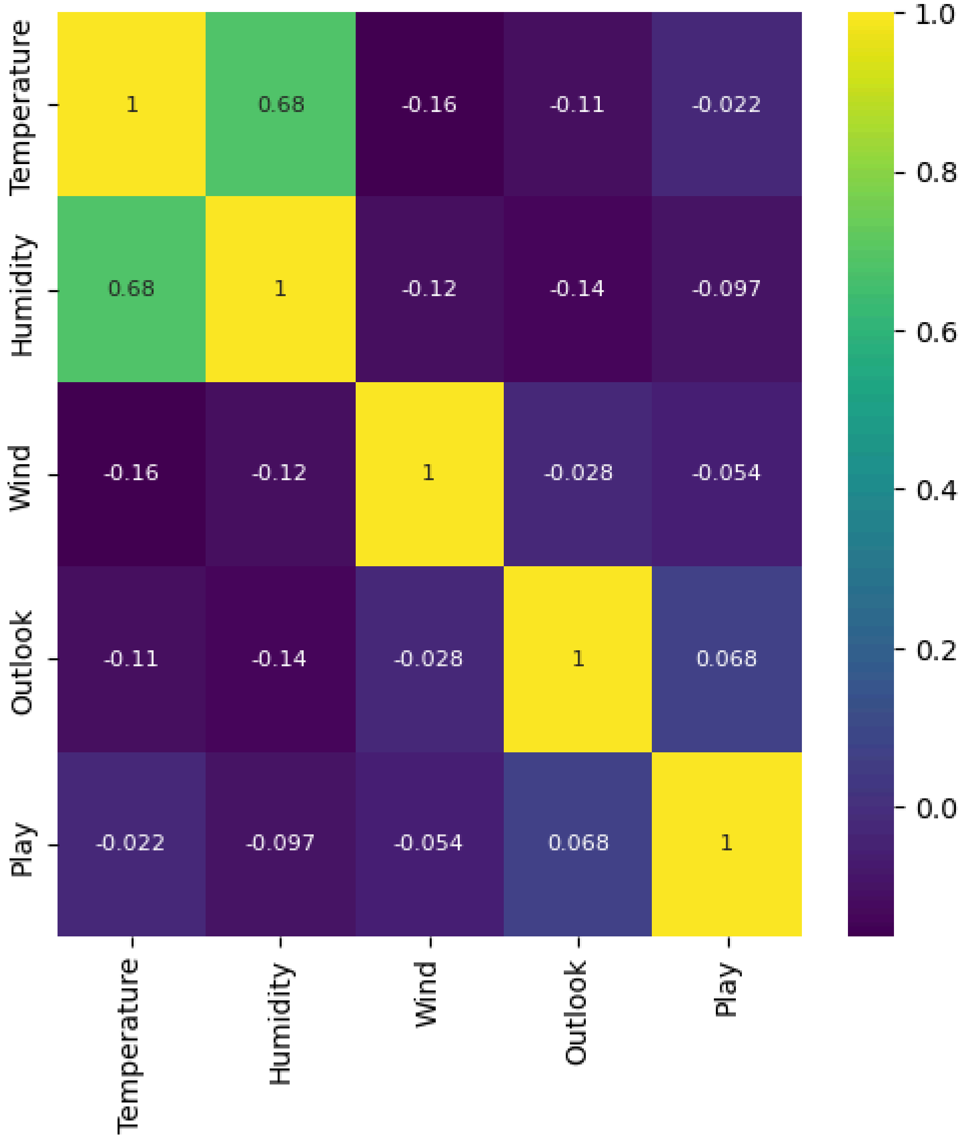
```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [104…

```
plt.figure(figsize=(6,6))
sns.heatmap(df[top_corr_features].corr(),annot=True,cmap="viridis", annot_kws={"fon
```

Out[104…

```
<Axes: >
```



In [105…

```
feature_cols= df.columns.drop(['Play'])
print(feature_cols)
```

```
Index(['Temperature', 'Humidity', 'Wind', 'Outlook'], dtype='object')
```

In [106…

```
feature_cols= ['Temperature','Humidity','Wind','Outlook']
print(feature_cols)
```

```
['Temperature', 'Humidity', 'Wind', 'Outlook']
```

In [107…

```
X = df[feature_cols]
X.head()
```

Out[107…

	Temperature	Humidity	Wind	Outlook
0	3.3	49	1	3
1	3.3	49	1	3
2	3.3	49	1	3
3	3.3	49	1	3
4	3.3	49	1	3

In [108…

```
y = df.Play
y.head()
```

Out[108…

```
0    1
1    0
2    0
3    1
4    1
Name: Play, dtype: int64
```

In [109…

```
from sklearn.model_selection import train_test_split
```

In [110…

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.45, random_
```

In [111…

```
len(y_train)
```

Out[111…

```
4215
```

In [112…

```
len(y_test)
```

Out[112…

```
3450
```

In [113…

```
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier(criterion = "entropy", max_depth=4)
```

In [114…

```
model= clf.fit(X_train, y_train)
```

In [115…

```
y_pred = model.predict(X_test)
```

In [116…

```
len(y_pred)
```

Out[116…

```
3450
```

In [117…

```
len(y_test)
```

Out[117…

```
3450
```

In [118…

```
y = pd.DataFrame({"Actual": y_test, "Predicted": y_pred})
y.head()
```

Out[118…

	Actual	Predicted
1940	0	0
1952	0	0
4755	0	0
5767	0	0
231	0	0

In [119…

```
y.tail()
```

Out[119…

	Actual	Predicted
1157	0	0
4626	0	0
632	0	0
1358	0	0
153	0	0

In [120…

```
y.sample(10)
```

Out[120…

	Actual	Predicted
4549	0	0
2779	0	0
537	0	0
4712	1	0
25	0	0
3291	1	0
5473	0	0
4361	0	0
5066	0	0
2761	0	0

In [121…

```
from sklearn import metrics
```

In [122…

```
c_mtrx = metrics.confusion_matrix(y_test, y_pred)
print("Confusion Matrix")
print(c_mtrx)
```

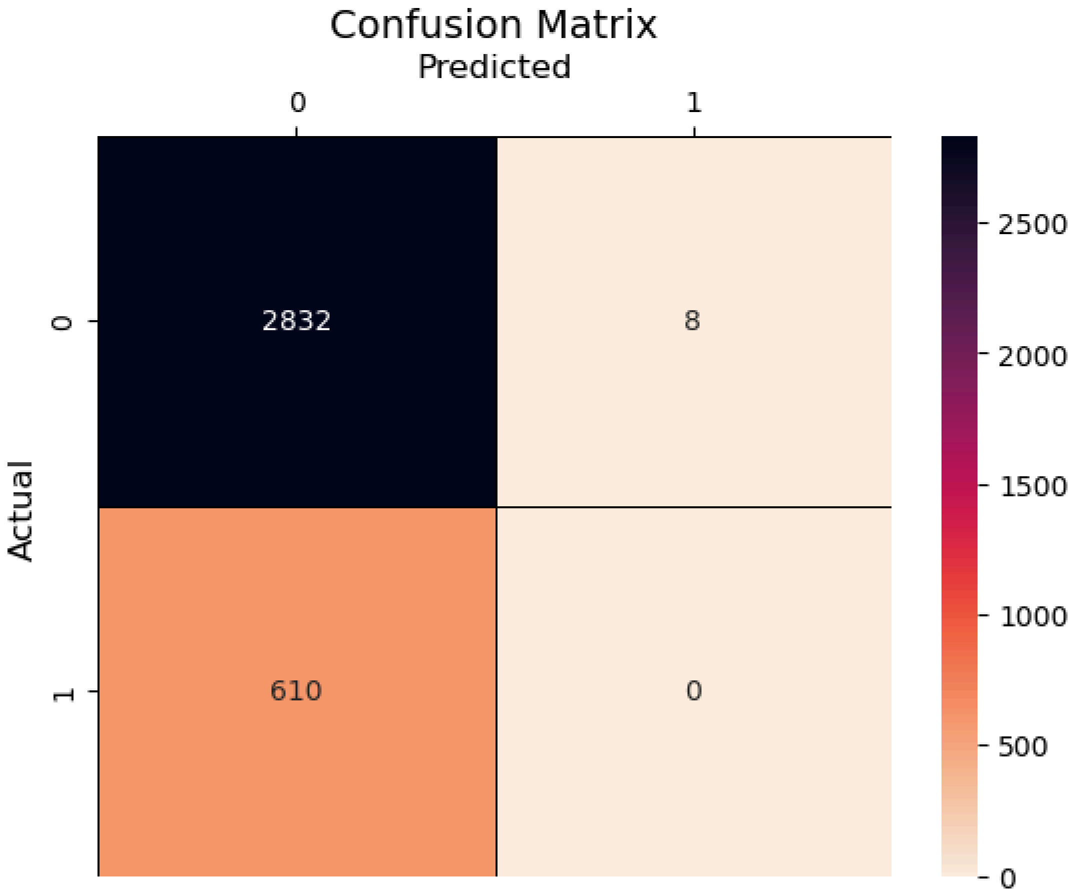
```
Confusion Matrix
[[2832    8]
 [ 610    0]]
```

In [123…

```
# Create the heatmap
ax = sns.heatmap(c_mtrx, annot=True, fmt='d', cbar=True, cmap="rocket_r", linewidth
# fmt='d' for integer format, using a colormap similar to the image

# Set predicted labels on top
ax.xaxis.tick_top() # Move the x-axis ticks (Predicted labels) to the top
ax.xaxis.set_label_position('top') # Move the x-axis label ('Predicted') to the top

# Set the axis labels and title
ax.set_xlabel('Predicted', fontsize=12)
ax.set_ylabel('Actual', fontsize=12)
ax.set_title('Confusion Matrix', fontsize=14)
```

Out[123…

```
Text(0.5, 1.0, 'Confusion Matrix')
```



In [124…

```
#[row, column]
#(Actual, Predict)
TN = c_mtrx[0, 0]
FP = c_mtrx[0, 1]
FN = c_mtrx[1, 0]
TP = c_mtrx[1, 1]

print("TN: ", TN, "\tFP: ", FP)
print("FN: ", FN, "\tTP: ", TP)
```

```
TN:  2832       FP:  8
FN:  610        TP:  0
```

In [125…

```
print('Metrics computed from a confusion matrix')
print("Accuracy:\t", metrics.accuracy_score(y_test, y_pred))
print("Sensitivity:\t", metrics.recall_score(y_test, y_pred))
print("Specificity:\t",TN / (TN + FP))
print("Precision:\t", metrics.precision_score(y_test, y_pred))
print("Classification Eerror:", 1 - metrics.accuracy_score(y_test, y_pred))
print("False_Positive_Rate:", 1 - TN / (TN + FP))
```

```
Metrics computed from a confusion matrix
Accuracy:        0.8208695652173913
Sensitivity:     0.0
Specificity:     0.9971830985915493
Precision:       0.0
Classification Eerror: 0.1791304347826087
False_Positive_Rate: 0.0028169014084507005
```

In [126…

```
count0 = df['Play'][df.Play == 0].count()

count1 = df['Play'][df.Play == 1].count()

print("Actual Dataset")
print("0's:",count0)
print("1's:",count1)
```

```
Actual Dataset
0's: 6266
1's: 1399
```

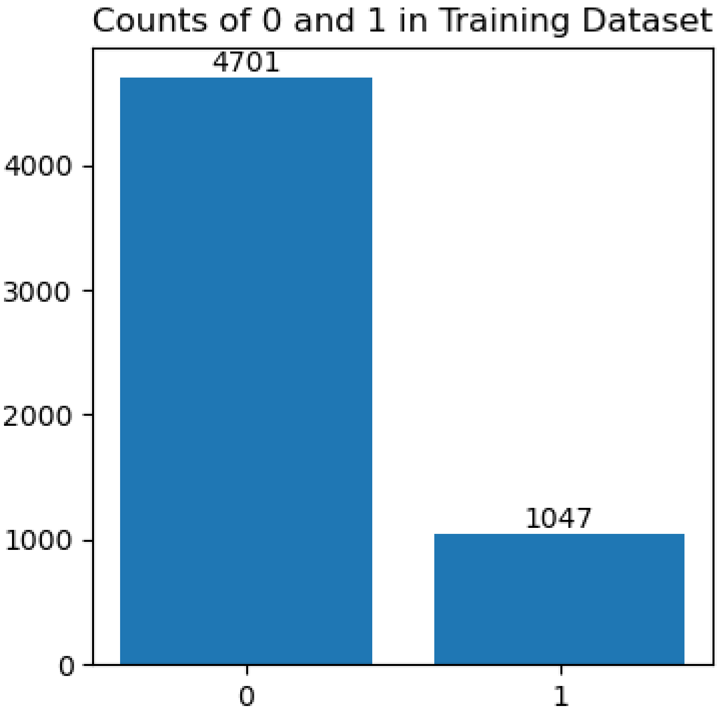
In [87]:

```
Trcount0 = sum(y_train==0)
Trcount1 = sum(y_train==1)

print("Trained Dataset")
print("0's:",Trcount0)
print("1's:",Trcount1)
```

```
Trained Dataset
0's: 4701
1's: 1047
```

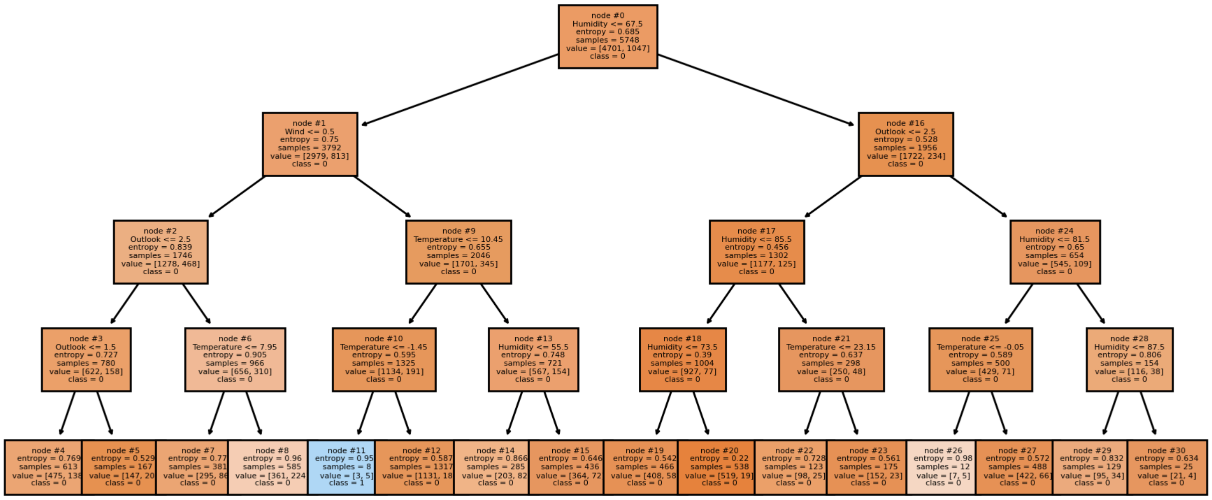
In [88]:

```
# Plotting the bar chart
labels = ['0', '1']
counts = [Trcount0, Trcount1]
plt.figure(figsize=(4,4))
plt.title('Counts of 0 and 1 in Training Dataset')
plt.bar(labels, counts)
# Add annotations to the bars
for i, count in enumerate(counts):
    plt.text(i, count, str(count), ha='center', va='bottom')

plt.show()
```



In [89]:
```
from sklearn import tree
```

In [90]:
```
# Plot the decision tree with customizations
plt.figure(figsize=(11, 5), dpi=200)
# plt.title("Decision Tree Visualization")
tree.plot_tree(clf, filled=True, feature_names=list(X_train.columns), class_names=[
                                   fontsize=4, max_depth=4, node_ids=True)

plt.show()
```


node #0 Humidity <= 67.5 entropy = 0.685 samples = 5748 value = [4701, 1047] class = 0
node #1 Wind <= 0.5 entropy = 0.75 samples = 3792 value = [2979, 813] class = 0
node #16 Outlook <= 2.5 entropy = 0.528 samples = 1956 value = [1722, 234] class = 0
node #2 Outlook <= 2.5 entropy = 0.839 samples = 1746 value = [1278, 468] class = 0
node #9 Temperature <= 10.45 entropy = 0.655 samples = 2046 value = [1701, 345] class = 0
node #17 Humidity <= 85.5 entropy = 0.456 samples = 1302 value = [1177, 125] class = 0
node #24 Humidity <= 81.5 entropy = 0.65 samples = 654 value = [545, 109] class = 0
node #3 Outlook <= 1.5 entropy = 0.727 samples = 780 value = [622, 158] class = 0
node #6 Temperature <= 7.95 entropy = 0.905 samples = 966 value = [656, 310] class = 0
node #10 Temperature <= -1.45 entropy = 0.595 samples = 1325 value = [1134, 191] class = 0
node #13 Humidity <= 55.5 entropy = 0.748 samples = 721 value = [567, 154] class = 0
node #18 Humidity <= 73.5 entropy = 0.39 samples = 1004 value = [927, 77] class = 0
node #21 Temperature <= 23.15 entropy = 0.637 samples = 298 value = [250, 48] class = 0
node #25 Temperature <= -0.05 entropy = 0.589 samples = 500 value = [429, 71] class = 0
node #28 Humidity <= 87.5 entropy = 0.806 samples = 154 value = [116, 38] class = 0
node #4 entropy = 0.769 samples = 613 value = [475, 138 class = 0
node #5 entropy = 0.529 samples = 167 value = [147, 20 class = 0
node #7 entropy = 0.77 samples = 381 value = [295, 86 class = 0
node #8 entropy = 0.96 samples = 585 value = [361, 224 class = 0
node #11 entropy = 0.95 samples = 8 value = [3, 5] class = 1
node #12 entropy = 0.587 samples = 1317 value = [1131, 18 class = 0
node #14 entropy = 0.866 samples = 285 value = [203, 82 class = 0
node #15 entropy = 0.646 samples = 436 value = [364, 72 class = 0
node #19 entropy = 0.542 samples = 466 value = [408, 58 class = 0
node #20 entropy = 0.22 samples = 538 value = [519, 19] class = 0
node #22 entropy = 0.728 samples = 123 value = [98, 25] class = 0
node #23 entropy = 0.561 samples = 175 value = [152, 23] class = 0
node #26 entropy = 0.98 samples = 12 value = [7, 5] class = 0
node #27 entropy = 0.572 samples = 488 value = [422, 66] class = 0
node #29 entropy = 0.832 samples = 129 value = [95, 34] class = 0
node #30 entropy = 0.634 samples = 25 value = [21, 4] class = 0

In [91]:
```
text_representation = tree.export_text(clf, feature_names=list(X_train.columns))
print(text_representation)
```

```
|--- Humidity <= 67.50
|   |--- Wind <= 0.50
|   |   |--- Outlook <= 2.50
|   |   |   |--- Outlook <= 1.50
|   |   |   |   |--- class: 0
|   |   |   |--- Outlook >  1.50
|   |   |   |   |--- class: 0
|   |   |--- Outlook >  2.50
|   |   |   |--- Temperature <= 7.95
|   |   |   |   |--- class: 0
|   |   |   |--- Temperature >  7.95
|   |   |   |   |--- class: 0
|   |--- Wind >  0.50
|   |   |--- Temperature <= 10.45
|   |   |   |--- Temperature <= -1.45
|   |   |   |   |--- class: 1
|   |   |   |--- Temperature >  -1.45
|   |   |   |   |--- class: 0
|   |   |--- Temperature >  10.45
|   |   |   |--- Humidity <= 55.50
|   |   |   |   |--- class: 0
|   |   |   |--- Humidity >  55.50
|   |   |   |   |--- class: 0
|--- Humidity >  67.50
|   |--- Outlook <= 2.50
|   |   |--- Humidity <= 85.50
|   |   |   |--- Humidity <= 73.50
|   |   |   |   |--- class: 0
|   |   |   |--- Humidity >  73.50
|   |   |   |   |--- class: 0
|   |   |--- Humidity >  85.50
|   |   |   |--- Temperature <= 23.15
|   |   |   |   |--- class: 0
|   |   |   |--- Temperature >  23.15
|   |   |   |   |--- class: 0
|   |--- Outlook >  2.50
|   |   |--- Humidity <= 81.50
|   |   |   |--- Temperature <= -0.05
|   |   |   |   |--- class: 0
|   |   |   |--- Temperature >  -0.05
|   |   |   |   |--- class: 0
|   |   |--- Humidity >  81.50
|   |   |   |--- Humidity <= 87.50
|   |   |   |   |--- class: 0
|   |   |   |--- Humidity >  87.50
|   |   |   |   |--- class: 0
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: