In [65]:
```
import pandas as pd
```

In [66]:
```
df = pd.read_csv("golf_dataset_long_format.csv")
df.head()
```

Out[66]:

	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

In [67]:
```
cols = list(df.columns)
print(cols)
```

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

In [68]:
```
df.shape
```

Out[68]:
```
(7665, 5)
```

In [69]:
```
corrmat = df.corr()
top_corr_features = corrmat.index
corrmat
```

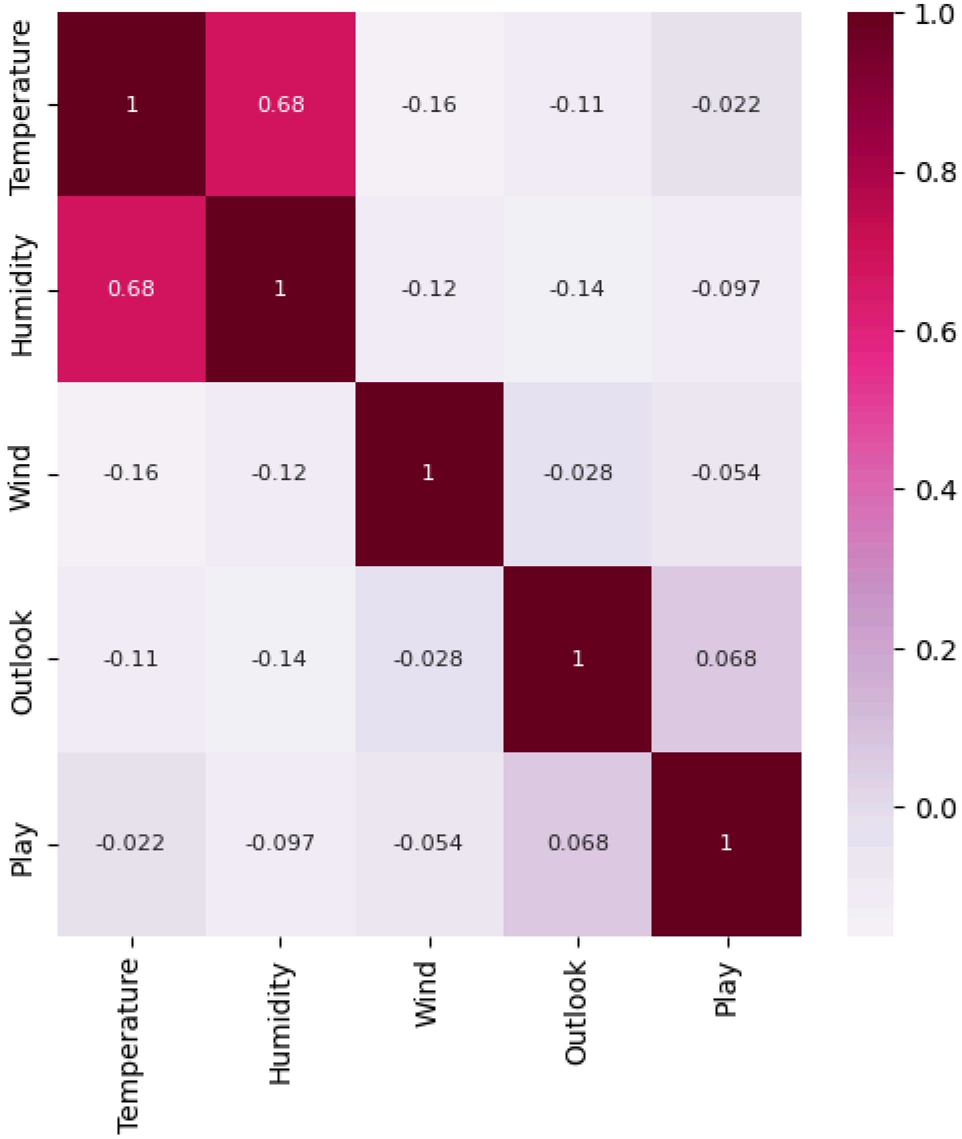
Out[69]:

	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 [70]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [71]:
```
plt.figure(figsize=(6,6))
sns.heatmap(df[top_corr_features].corr(),annot=True,cmap="PuRd", annot_kws={"fontsi
```

Out[71]:
```
<Axes: >
```



In [72]:
```
feature_cols= df.columns.drop(['Play'])
print(feature_cols)
```

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

In [73]:
```
X = df[feature_cols]
X.head()
```

Out[73]:

	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 [74]:

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

Out[74]:

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

In [75]:

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

In [76]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_
```

In [77]:

```
len(y_train)
```

Out[77]:

```
5748
```

In [78]:

```
len(y_test)
```

Out[78]:

```
1917
```

In [79]:

```
from sklearn.naive_bayes import GaussianNB
clf = GaussianNB()
```

In [80]:

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

In [81]:

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

In [82]:

```
len(y_pred)
```

Out[82]:

```
1917
```

In [83]:

```
len(y_test)
```

Out[83]:

```
1917
```

In [84]:

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

Out[84]:

	Actual	Predicted
785	1	0
3806	0	0
6333	0	0
193	1	0
1986	0	0

In [85]:
```python
y.tail()
```

Out[85]:

	Actual	Predicted
6553	0	0
4825	0	0
4514	0	0
5003	0	0
5485	0	0

In [86]:
```python
y.sample(10)
```

Out[86]:

	Actual	Predicted
4953	0	0
3594	0	0
127	0	0
1236	1	0
6977	0	0
2515	0	0
7619	0	0
1012	0	0
5974	0	0
5079	1	0

In [87]:
```python
from sklearn import metrics
```

In [88]:
```python
c_mtrx = metrics.confusion_matrix(y_test, y_pred)
print("Confusion Matrix")
print(c_mtrx)
```

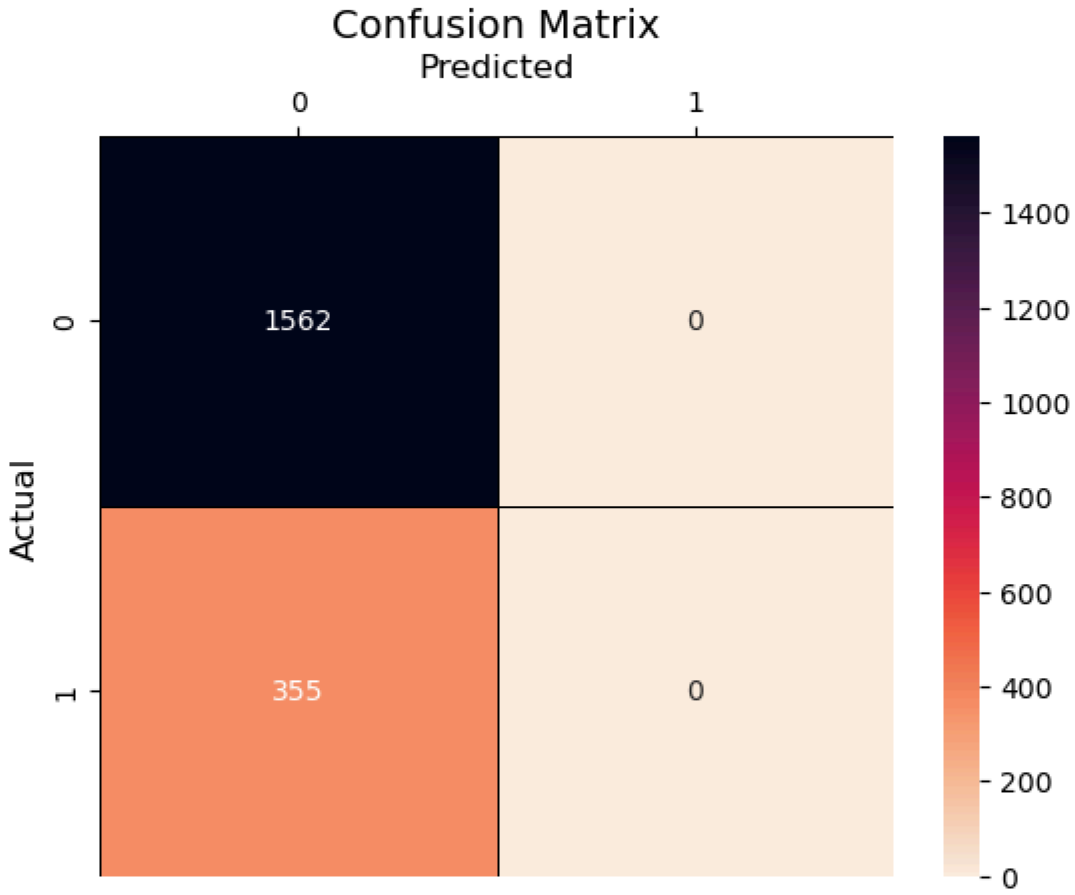
```
Confusion Matrix
[[1562    0]
 [ 355    0]]
```

In [89]:
```python
# 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[89]:

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



In [90]:

```
#[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:  1562       FP:  0
FN:  355        TP:  0
```

In [91]:

```
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 Error:", 1 - metrics.accuracy_score(y_test, y_pred))
print("False_Positive_Rate:", 1 - TN / (TN + FP))
```

```
Metrics computed from a confusion matrix
Accuracy:        0.8148148148148148
Sensitivity:     0.0
Specificity:     1.0
Precision:       0.0
Classification Eerror: 0.18518518518518523
False_Positive_Rate: 0.0
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1469:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 due to no pred
icted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [92]:

```
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
```

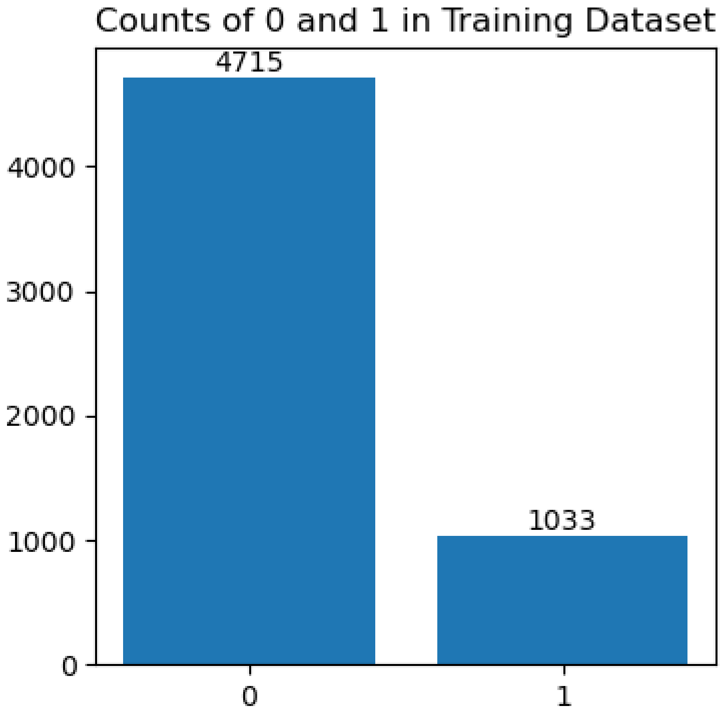
In [60]:

```
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: 4715
1's: 1033
```

In [61]:

```
# 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()
```


Counts of 0 and 1 in Training Dataset
4715
1033
0
1000
2000
3000
4000
0
1

```
In [ ]:
```

```
In [ ]:
```