```
# Name:- Rohit Kumar
# Roll_No:- 25901334
# Model:- ID3 Decision Tree Regression
```

```
import numpy as np
import pandas as pd
```

```
# Create dataset as pandas DataFrame
data = pd.DataFrame({
    'Outlook': ['Sunny','Sunny','Overcast','Rain','Rain','Rain','Overcast','Sunny','Sunny','Rain
    'Temperature': ['Hot','Hot','Hot','Mild','Cool','Cool','Cool','Mild','Cool','Mild','Mild','Mi
    'Humidity': ['High','High','High','High','Normal','Normal','Normal','High','Normal','Normal',
    'Windy': [False, True, False, False, False, True, True, False, False, False, True, True, Fals
    'Play': ['No','No','Yes','Yes','Yes','No','Yes','No','Yes','Yes','Yes','Yes','Yes','No']
})

data.head()
```

	Outlook	Temperature	Humidity	Windy	Play
0	Sunny	Hot	High	False	No
1	Sunny	Hot	High	True	No
2	Overcast	Hot	High	False	Yes
3	Rain	Mild	High	False	Yes
4	Rain	Cool	Normal	False	Yes

Next steps: Generate code with data | New interactive sheet

```
def entropy(target_col):
    elements, counts = np.unique(target_col, return_counts=True)
    entropy = -np.sum([(counts[i]/np.sum(counts))*np.log2(counts[i]/np.sum(counts)) for i in rang
    return entropy

# Entropy of entire dataset
total_entropy = entropy(data['Play'])
print("Entropy of Play:", total_entropy)
```

```
Entropy of Play: 0.9402859586706311
```

```
def InfoGain(data, split_attribute_name, target_name="Play"):
    # Total entropy
    total_entropy = entropy(data[target_name])

    # Values and counts
    vals, counts = np.unique(data[split_attribute_name], return_counts=True)

    # Weighted entropy
    Weighted_Entropy = np.sum([(counts[i]/np.sum(counts)) *
                               entropy(data.where(data[split_attribute_name]==vals[i]).dropna()[ta
                               for i in range(len(vals))])

    # Information Gain
    Information_Gain = total_entropy - Weighted_Entropy
    return Information_Gain

# Compute IG for all features
for col in ['Outlook','Temperature','Humidity','Windy']:
    print(f"Information Gain for {col}:", InfoGain(data, col))
```

```
Information Gain for Outlook: 0.24674981977443933
Information Gain for Temperature: 0.02922256565895487
Information Gain for Humidity: 0.15183550136234159
Information Gain for Windy: 0.04812703040826949
```

```
def predict(row):
    if row['Outlook']=='Overcast':
        return 'Yes'
    elif row['Outlook']=='Sunny':
        return 'Yes' if row['Humidity']=='Normal' else 'No'
    elif row['Outlook']=='Rain':
        return 'Yes' if row['Windy']==False else 'No'

data['Predicted'] = data.apply(predict, axis=1)
accuracy = np.mean(data['Play'] == data['Predicted'])
print("Predicted classes:", data['Predicted'].values)
print(f"Accuracy: {accuracy*100:.2f}%")
```

```
Predicted classes: ['No' 'No' 'Yes' 'Yes' 'Yes' 'No' 'Yes' 'No' 'Yes' 'Yes' 'Yes' 'Yes' 'Yes'
 'No']
Accuracy: 100.00%
```