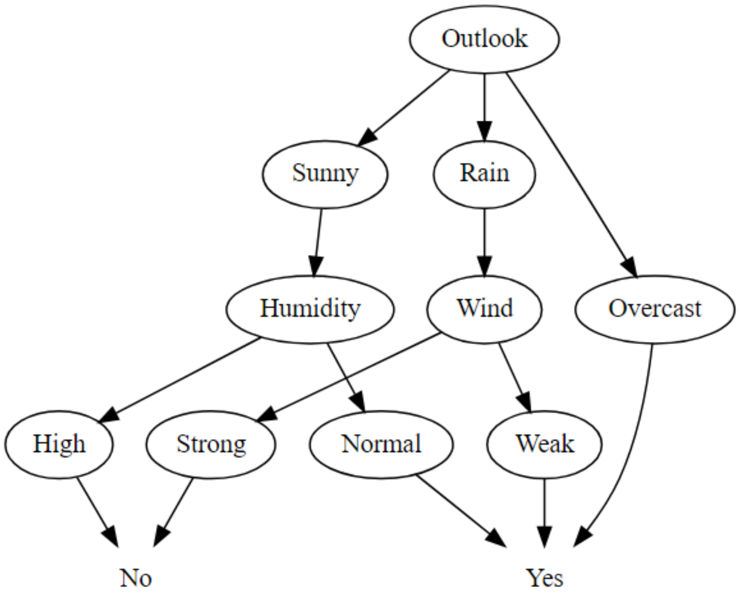
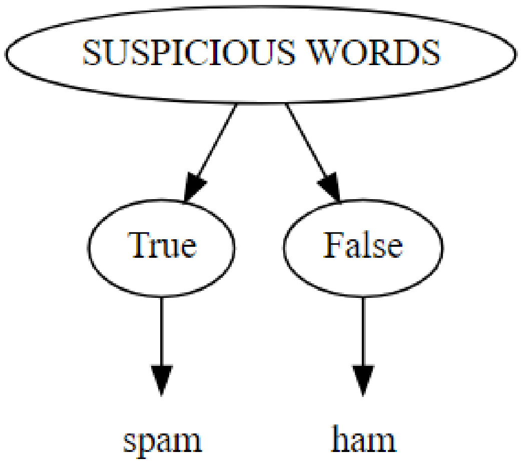
[50]:
```python
from graphviz import Digraph

def draw_decision_tree_dictionary(tree_dictionary):
    if not isinstance(tree_dictionary, dict):
        raise TypeError("Argument must be of type dictionary")
    if not tree_dictionary:
        raise ValueError("Dictionary tree_dictionary is empty")

    dot = Digraph(strict=True)
    draw_tree(dot, tree_dictionary,None)

    return dot

def draw_tree(dot, tree_dictionary, parent_node_name):
    if isinstance(tree_dictionary, dict):
        for key in tree_dictionary:
            no_spaces_key = str(key).replace(" ","")

            dot.node(no_spaces_key, str(key), shape="ellipse")

            if parent_node_name != None:
                dot.edge(parent_node_name, no_spaces_key)

            draw_tree(dot, tree_dictionary[key], no_spaces_key)

    else:
        val = str(tree_dictionary)
        dot.node(val, val,shape="plaintext")
        dot.edge(parent_node_name, val)

dd = draw_decision_tree_dictionary(model)
dd
```

[50]:



[54]:
```python
from graphviz import Digraph

def draw_decision_tree_dictionary(tree_dictionary):
    if not isinstance(tree_dictionary, dict):
        raise TypeError("Argument must be of type dictionary")
    if not tree_dictionary:
        raise ValueError("Dictionary tree_dictionary is empty")

    dot = Digraph(strict=True)
    draw_tree(dot, tree_dictionary,None)

    return dot

def draw_tree(dot, tree_dictionary, parent_node_name):
    if isinstance(tree_dictionary, dict):
        for key in tree_dictionary:
            no_spaces_key = str(key).replace(" ","")

            dot.node(no_spaces_key, str(key), shape="ellipse")

            if parent_node_name != None:
                dot.edge(parent_node_name, no_spaces_key)

            draw_tree(dot, tree_dictionary[key], no_spaces_key)

    else:
        val = str(tree_dictionary)
        dot.node(val, val,shape="plaintext")
        dot.edge(parent_node_name, val)

dd = draw_decision_tree_dictionary(model)
dd
```

[54]:



Census Dataset Without Pruning

```
[13]: test = pd.read_csv("assets/census_training_test.csv")

      correct = 0
      incorrect = 0

      def predict(query, model):

          if not isinstance(model, dict):
              return model

          feature = next(iter(model))
          feature_value = query.get(feature)

          subtree = model[feature].get(feature_value)

          if subtree is None:
              return None

          return predict(query, subtree)


      for k,row in test.iterrows():
          prediction = predict(row, model)

          if prediction == row.iloc[-1]:
              correct += 1
          else:

              incorrect += 1


      print("Number of testing examples =  ", incorrect +correct)
      print("correct_classification_count =  " , correct )
      print("incorrect_classification_count = " , incorrect)
      print("accuracy = ", correct/(incorrect+correct) * 100)
```

```
Number of testing examples =   15028
correct_classification_count =   12110
incorrect_classification_count =  2918
accuracy =  80.5829118977908
```

With Pruning

[19]:

```python
test = pd.read_csv("assets/census_training_test.csv")

correct = 0
incorrect = 0

def predict(query, model):

    if not isinstance(model, dict):
        return model

    feature = next(iter(model))
    feature_value = query.get(feature)

    subtree = model[feature].get(feature_value)

    if subtree is None:
        return None

    return predict(query, subtree)


for k,row in test.iterrows():
    prediction = predict(row, model)

    if prediction == row.iloc[-1]:
        correct += 1
    else:

        incorrect += 1


print("Number of testing examples =  ", incorrect +correct)
print("correct_classification_count =  " , correct )
print("incorrect_classification_count = " , incorrect)
print("accuracy = ", correct/(incorrect+correct) * 100)
```

```
Number of testing examples =   15028
correct_classification_count =   12284
incorrect_classification_count =  2744
accuracy =  81.74075059888209
```