Report on Bayesian Network Model for Alarm System Overview

Introduction:

The provided code implements a Bayesian Network model using the pgmpy library, focusing on modularity and extensibility. The Bayesian Network represents an alarm system, where events such as burglary, earthquake, alarm activation, and phone calls are modeled along with their conditional dependencies.

Code Structure:

1. **Library Imports:** The code begins with the import of necessary libraries, such as numpy, pandas, and pgmpy. The purpose of each library is briefly explained.

```
from pgmpy.models import BayesianNetwork
from pgmpy.inference import VariableElimination
```

2. **File Exploration:** The code utilizes the os module to explore the contents of the '/kaggle/input/' directory. A loop is used to print the file names within the directory.

3. **pgmpy Installation:** The code installs the pgmpy library using the !pip install pgmpy command.

```
!pip install pgmpy
```

4. **Bayesian Network Structure:** The Bayesian Network is defined using the BayesianNetwork class from pgmpy.models. The network structure involves nodes representing events like burglary, earthquake, alarm, John calling, and Mary calling. Dependencies among these events are specified as directed edges in the network.

```
alarm_model = BayesianNetwork( [
        ("Burglary", "Alarm"),
        ("Earthquake", "Alarm"),
        ("Alarm", "JohnCalls"),
        ("Alarm", "MaryCalls"),] )
```

5. **Conditional Probability Distributions (CPDs):** Conditional Probability Distributions for each node are defined using the TabularCPD class from pgmpy.factors.discrete. Probability values for different scenarios are specified for each node, taking into account evidence nodes.

```
from pgmpy.factors.discrete import TabularCPD
cpd_burglary = TabularCPD(
    variable="Burglary", variable_card=2, values=[[0.999], [0.001]]
)
cpd_earthquake = TabularCPD(
    variable="Earthquake", variable_card=2, values=[[0.998], [0.002]]
)

cpd_alarm = TabularCPD(
    variable="Alarm",
    variable_card=2,
    values=[[0.999, 0.71, 0.06, 0.05], [0.001, 0.29, 0.94, 0.95]],
    evidence=["Burglary", "Earthquake"],
    evidence_card=[2, 2],
)
cpd_johncalls = TabularCPD(
    variable="JohnCalls",
    variable_card=2,
    values=[[0.95, 0.1], [0.05, 0.9]],
    evidence=["Alarm"],
    evidence_card=[2],
)
cpd_marycalls = TabularCPD(
    variable="MaryCalls",
    variable_card=2,
    values=[[0.1, 0.7], [0.9, 0.3]],
    evidence=["Alarm"],
    evidence_card=[2],
)
```

6. **Associating CPDs with the Model:** The defined CPDs are associated with the model structure using the add_cpds method.

```
alarm_model.add_cpds(
    cpd_burglary, cpd_earthquake, cpd_alarm, cpd_johncalls, cpd_marycalls
)
```

7. **Model Validation:** The check_model method is used to ensure that the CPDs are valid for the specified model structure.

```
alarm_model.check_model()
```

8. **Model Exploration:** The code includes functions to view the nodes and edges of the model. It also checks the local independencies of a specific node ('Burglary') and lists all independencies in the model.

```
alarm_model.nodes()
alarm_model.edges()
alarm_model.local_independencies("Burglary")
alarm_model.get_independencies()
```

Outcome:

For listing all independencies:

```
(Earthquake ⊥ Burglary)
(Earthquake ⊥ JohnCalls, MaryCalls | Alarm)
(Earthquake ⊥ MaryCalls | Alarm, JohnCalls)
(Earthquake ⊥ JohnCalls, MaryCalls | Alarm, Burglary)
(Earthquake ⊥ JohnCalls | Alarm, MaryCalls)
(Earthquake ⊥ MaryCalls | Alarm, Burglary, JohnCalls)
(Earthquake ⊥ JohnCalls | Alarm, Burglary, MaryCalls)
(JohnCalls ⊥ Earthquake, Burglary, MaryCalls | Alarm)
(JohnCalls ⊥ Burglary, MaryCalls | Alarm, Earthquake)
(JohnCalls ⊥ Earthquake, MaryCalls | Alarm, Burglary)
(JohnCalls ⊥ Earthquake, Burglary | Alarm, MaryCalls)
(JohnCalls ⊥ MaryCalls | Alarm, Earthquake, Burglary)
(JohnCalls ⊥ Burglary | Alarm, Earthquake, MaryCalls)
(JohnCalls ⊥ Earthquake | Alarm, Burglary, MaryCalls)
(Burglary ⊥ Earthquake)
(Burglary ⊥ JohnCalls, MaryCalls | Alarm)
(Burglary ⊥ JohnCalls, MaryCalls | Alarm, Earthquake)
(Burglary ⊥ MaryCalls | Alarm, JohnCalls)
(Burglary ⊥ JohnCalls | Alarm, MaryCalls)
(Burglary ⊥ MaryCalls | Alarm, Earthquake, JohnCalls)
(Burglary ⊥ JohnCalls | Alarm, Earthquake, MaryCalls)
(MaryCalls ⊥ Earthquake, JohnCalls, Burglary | Alarm)
(MaryCalls ⊥ JohnCalls, Burglary | Alarm, Earthquake)
(MaryCalls ⊥ Earthquake, Burglary | Alarm, JohnCalls)
(MaryCalls ⊥ Earthquake, JohnCalls | Alarm, Burglary)
(MaryCalls ⊥ Burglary | Alarm, Earthquake, JohnCalls)
(MaryCalls ⊥ JohnCalls | Alarm, Earthquake, Burglary)
(MaryCalls ⊥ Earthquake | Alarm, Burglary, JohnCalls)
```

Conclusion:

The code successfully implements a Bayesian Network model for an alarm system using the pgmpy library. This model can be utilized for probabilistic reasoning and inference about the occurrences of events in the alarm system based on observed evidence. The clarity in code organization and the use of the pgmpy library contribute to the modularity and extensibility of the implementation.