Summary

A Bayesian model for Credit Worthiness based on German Dataset

Jyoti Yadav, jyoti.yadav@studio.unibo.it
Fundamentals of Artificial Intelligence and Knowledge Representation (Mod. 3)
Alma Mater Studiorum Università di Bologna

1. Introduction

This is a Bayesian model to learn the association between the traits and attributes of different borrowers in history and their repayment status i.e. whether it resulted in Good Risk or Bad Risk. It is the risk of witnessing defaults on a debt that may arise from a borrower failing to make required payments

In this we will use pgmpy [pgmpy] is a python library for working with graphical models. It allows the user to create their own graphical models and answer inference or map queries over them. pgmpy has implementation of many inference algorithms like VariableElimination, Belief Propagation etc.

We can install the pgmpy library using the simple command

```
!pip install pgmpy
```

2. Reading CSV file

A dataset of 1000 rows and 10 columns. There are 9 input features Credit History, Age, Gender, Job ,Housing, Savings accounts, Credit amount, Duration, Purpose . We have the target variable as good vs Bad Risk.

Column	Values
Credit History	0 to 4

Age	19 to 75
Gender	Male and Female
Job	0 to 3
Housing	Own, rent, and free
Savings accounts	Little, nan, quite,rich,moderate
Credit Amount	250 to 15945
Duration	4 to 72
Purpose	Furniture/equipment, radio/TV, business,education,repairs,domestic/ appliances, vacation/others
Risk	Good, bad

We will read the csv file using the pandas function read.csv()

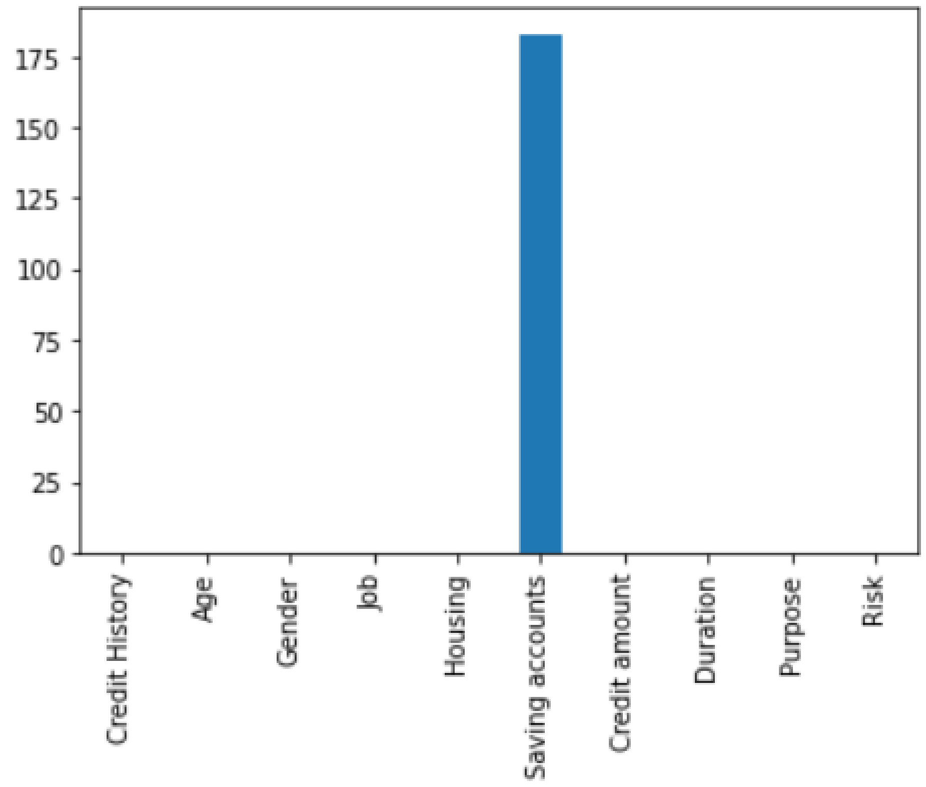
```
df=pd.read_csv('/content/german_credit.csv')
```

3. Data cleaning

We will start with data cleaning and the first thing will be to check the nan values

```
df.isnull().sum().plot(kind='bar')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fca6f0e5c50>
```



The savings account has NaN entries so here I decided to replace the nan values with another category called ‘no accounts’.

We also have to made certain assumptions and we decided to bin the numerical features into three categories which have a numerical values greater than 32 because we wanted to check if young people are more credit worthy

```
col_list=[]
for col in df.columns:
  print(col,":",len(df[col].unique()))
  if(len(df[col].unique())>32):
    col_list.append(col)
```

Let's make the final dataframe a little cleaner to understand the range of values it can take based on its min and max value
Age has min and max as 19 and 75 respectively

```
min(df['Age'].unique()), max(df['Age'].unique())
bin_age = ['19 to 28', '28 to 38', '38 to 75']
```

Credit amount has min and max as 250 and 18424 respectively

```
min(df['Credit amount'].unique()), max(df['Credit amount'].unique())
bin_credit_amount = ['250 to 1554', '1554 to 3368', '3368 to 18424']
```

Duration has min and max as 4 and 72 respectively

```
min(df.Duration.unique()), max(df.Duration.unique())
bin_duration = ['4 to 12', '12 to 24', '24 to 72']
```

4. Train and Test Split

We divide the data into train and test
We take 80% as raining data and 20% as testing data with the target as Risk .
We have intotal 1000 rows and we take 800 as training data and 200 as testing data

```
np.random.seed(50)
sample = np.random.choice(df.index, size=int(len(df) * 0.8), replace=False)
#complete data
print(df.shape)

#Training data
train_data = df.iloc[sample]
print(train_data.shape)

#Testing data
test_data = df.drop(sample)
print(test_data.shape)

#Testing target
test_y = test_data['Risk']
print(test_y.shape)

#Testing features
test_data.drop(columns=['Risk'], inplace=True)
print(test_data.shape)
```

5. Model

When I decided to Model my problem I came across two models, the Bayesian Model and the HillClimbSearch each Model.I decided to model this problem with two models. In order to check the accuracy which performs better we will use it for the inference.

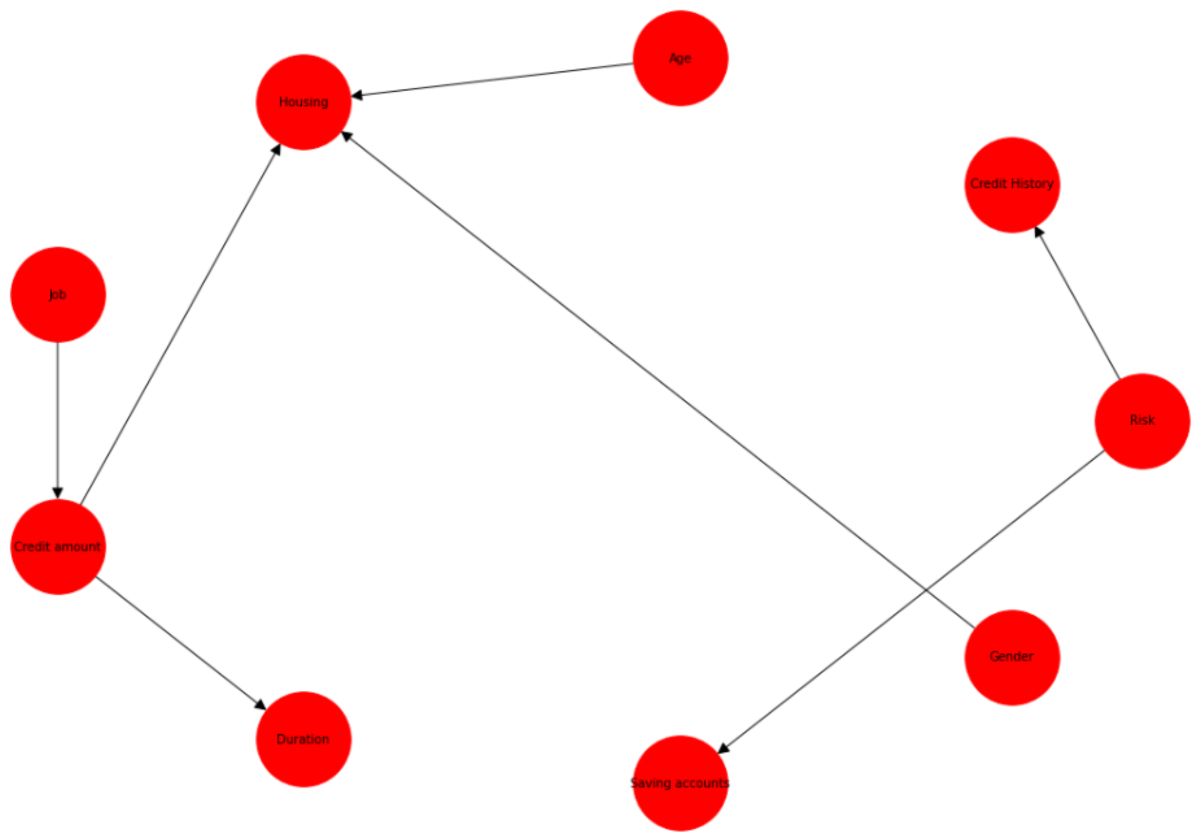
Bayesian Model

A Bayesian Network consists of a directed graph where nodes represent random variables and edges represent the relation between them. It is parameterized using Conditional Probability Distributions(CPD). Each random variable in a Bayesian Network has a CPD associated with it. If a random variable has parents in the network then the CPD represents P(var|Parvar) i.e. the probability of that variable given its parents. In the case, when the random variable has no parents it simply represents P(var) i.e. the probability of that variable

Based on the analysis i define the network structure

```
#My custom model

model11=BayesianModel([('Risk','Credit History'),
                      ('Age','Housing'),
                      ('Job','Credit amount'),
                      ('Credit amount','Housing'),
                      ('Credit amount','Duration'),
                      ('Risk','Saving accounts'),
                      ('Gender','Housing')])

print('Nodes',model11.nodes(),'\n')
print('Edges',model11.edges())

plt.figure(figsize=(15,10))
nx.draw_circular(model11, with_labels=True, arrowsize=20, node_size=6000,
font_size=10,node_color='red')
plt.show()
```

Visualizing the Bayesian Network



Fitting the model11 with the bayesian estimator

Bayesian Estimator - Method to estimate the CPD for a given variable.
BDeu - Computes a score that measures how much a given variable is “influenced” by a given list of potential parents.
equivalent_sample_size=10
complete_samples_only=False

```
model11.fit(data=df,estimator=BayesianEstimator,prior_type="BDeu",equivalent_sa
mple_size=10,complete_samples_only=False)
```

2. HillClimbSearch

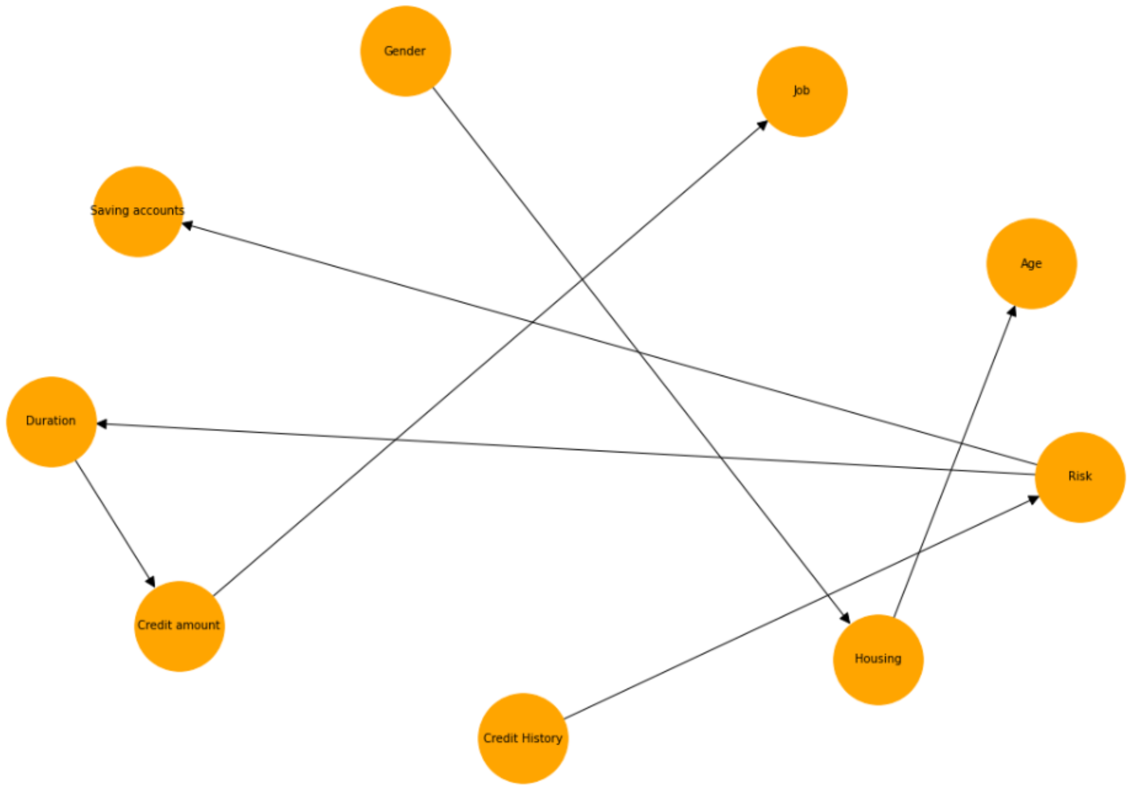
Hill climbing algorithm is a local search algorithm which continuously moves in the direction of increasing elevation/value to find the peak of the mountain or best solution to the problem. It terminates when it reaches a peak value where no neighbor has a higher value.

HillClimbSearch in pgmpy - Performs local hill climb search to estimate the DAG structure that has optimal score, according to the scoring method supplied. Starts at model start_dag and proceeds by step-by-step network modifications until a local maximum is reached. Only estimates network structure, no parametrization.
It returns a model Returns model – A DAG at a (local) score maximum.
Here i have used BicScore to generate a model

Performs local hill climb search to estimate the DAG structure that has optimal score, according to the scoring method supplied. Starts at model start_dag and proceeds by step-by-step network modifications until a local maximum is reached. Only estimates network structure, no parametrization.

BicScore - The score to be optimized during structure estimation .Computes a score that measures how much a given variable is "influenced" by a given list of potential parents.

```
model2 = HillClimbSearch(train_data)
best_model = model2.estimate(scoring_method=BicScore(train_data))
edges = list(best_model.edges())
model22 = BayesianModel(edges)

plt.figure(figsize=(15,10))
nx.draw_circular(model22, with_labels=True, arrowsize=20, node_size=6000,
font_size=10,node_color='orange')
plt.show()
```



Here I fit the model with MaximumLikelihoodEstimatorIt.
Method to estimate the model parameters (CPDs) using Maximum Likelihood Estimation

```
model22.fit(train_data, estimator=MaximumLikelihoodEstimator)
```

6.Prediction

```
columns = list(model11.nodes)
columns.remove('Risk')

y_pred = model11.predict(test_data[columns])['Risk']
# evaluate predictions
acc = accuracy_score(test_y, y_pred)
print('Accuracy Bayesian Model : %.3f' % acc)

f1=f1_score(test_y, y_pred, average=None)
print('f1 score is',f1)
```

```
Accuracy Bayesian Model : 0.775
f1 score is [0.38356164 0.86238532]
```

```
columns = list(model22.nodes)
columns.remove('Risk')

y_pred = model22.predict(test_data[columns])['Risk']
# evaluate predictions
acc = accuracy_score(test_y, y_pred)
print('Accuracy HillSearchModel : %.3f' % acc)

f1=f1_score(test_y, y_pred, average=None)
print('f1 score is',f1)
```

```
Accuracy HillSearchModel : 0.775
f1 score is [0.46315789 0.83278689]
```

7. Conditional Probability Distribution

We have a method get_cpds() .Here we define the CPT for each random variable

```
l1=model11.get_cpds()
for cpd in l1:
    print(f'CPT of {cpd.variable}:')
    print(cpd, '\n')
```

Conditional Probability distribution for Risk

```
 CPT of Risk:

+------------+---------+

| Risk(bad)  | 0.30198 |

+------------+---------+

| Risk(good) | 0.69802 |

+------------+---------+
```

Conditional Probability distribution for Credit History

```
CPT of Credit History:

+-------------------+---------------------+----------------------+

| Risk              | Risk(bad)           | Risk(good)           |

+-------------------+---------------------+----------------------+

| Credit History(0) | 0.08524590163934426 | 0.02269503546099291  |

+-------------------+---------------------+----------------------+

| Credit History(1) | 0.09508196721311475 | 0.031205673758865248 |

+-------------------+---------------------+----------------------+

| Credit History(2) | 0.5573770491803278  | 0.5134751773049645   |

+-------------------+---------------------+----------------------+

| Credit History(3) | 0.09508196721311475 | 0.08652482269503546  |

+-------------------+---------------------+----------------------+

| Credit History(4) | 0.16721311475409836 | 0.34609929078014184  |

+-------------------+---------------------+----------------------+
```

Conditional Probability distribution for Age

```
CPT of Age:

+---------------+-----------+

| Age(19 to 28) | 0.652805  |

+---------------+-----------+

| Age(28 to 38) | 0.276568  |

+---------------+-----------+

| Age(38 to 75) | 0.0706271 |

+---------------+-----------+
```

Conditional Probability distribution for Housing

```
+----------------+-----+-------------------------------+
| Age            | ... | Age(38 to 75)                 |
+----------------+-----+-------------------------------+
| Credit amount  | ... | Credit amount(3368 to 18424)  |
+----------------+-----+-------------------------------+
| Gender         | ... | Gender(male)                  |
+----------------+-----+-------------------------------+
| Housing(free)  | ... | 0.4796747967479675            |
+----------------+-----+-------------------------------+
| Housing(own)   | ... | 0.26016260162601623           |
+----------------+-----+-------------------------------+
| Housing(rent)  | ... | 0.26016260162601623           |
+----------------+-----+-------------------------------+
```

Conditional Probability distribution for Job

```
+--------+-----------+
| Job(0) | 0.0242574 |
+--------+-----------+
| Job(1) | 0.200495  |
+--------+-----------+
| Job(2) | 0.626238  |
+--------+-----------+
| Job(3) | 0.14901   |
+--------+----------+
```

Conditional Probability distribution for Credit amount

```
+-------------------------------+-----+---------------------+
| Job                           | ... | Job(3)              |
+-------------------------------+-----+---------------------+
| Credit amount(1554 to 3368)   | ... | 0.31118493909191586 |
+-------------------------------+-----+---------------------+
| Credit amount(250 to 1554)    | ... | 0.6035437430786268  |
+-------------------------------+-----+---------------------+
| Credit amount(3368 to 18424)  | ... | 0.08527131782945736 |
+-------------------------------+-----+---------------------+
```

Conditional Probability distribution for Duration

```
+--------------------+-----+------------------------------+
| Credit amount      | ... | Credit amount(3368 to 18424) |
+--------------------+-----+------------------------------+
| Duration(12 to 24) | ... | 0.4527363184079602           |
+--------------------+-----+------------------------------+
| Duration(24 to 72) | ... | 0.2736318407960199           |
+--------------------+-----+------------------------------+
| Duration(4 to 12)  | ... | 0.2736318407960199           |
+--------------------+-----+------------------------------+
```

Conditional Probability distribution for savings accounts

```
+-------------------------------+----------------------+--------------------
--+

| Risk                          | Risk(bad)            | Risk(good)
|

+-------------------------------+----------------------+--------------------
--+

| Saving accounts(little)       | 0.7147540983606557   | 0.548936170212766
|

+-------------------------------+----------------------+--------------------
--+

| Saving accounts(moderate)     | 0.11475409836065574  |
0.09929078014184398 |

+-------------------------------+----------------------+--------------------
--+

| Saving accounts(no accounts) | 0.10819672131147541  |
0.21560283687943263 |

+-------------------------------+----------------------+--------------------
--+

| Saving accounts(quite rich)  | 0.03934426229508197  | 0.075177304964539
|

+-------------------------------+----------------------+--------------------
--+

| Saving accounts(rich)         | 0.022950819672131147 |
0.06099290780141844 |

+-------------------------------+----------------------+--------------------
-+
```

Conditional Probability distribution for Gender

```
+----------------+----------+

| Gender(female) | 0.311881 |
+----------------+----------+
| Gender(male)    | 0.688119 |
+----------------+----------+
```

8. Independence

```
for i in model11.nodes():
  print(" Independencies :",i,'\n',model11.local_independencies(i))
  print(" Descendants :",i,'\n',model11.get_children(i))
  print('\n')
```

Independence/Descendents of Risk

```
Independencies : Risk
 (Risk ⟂ Housing, Credit amount, Age, Duration, Job, Gender)
 Descendants : Risk
 ['Credit History', 'Saving accounts']
```

Independence/Descendents of Credit History

```
 Independencies : Credit History
 (Credit History ⟂ Housing, Credit amount, Age, Saving accounts, Duration, Job,
Gender | Risk)
 Descendants : Credit History
 []
```

Independence/Descendents of Age

```
 Independencies : Age
 (Age ⟂ Risk, Saving accounts, Credit amount, Credit History, Duration, Job,
Gender)
 Descendants : Age
 ['Housing']
```

Independence/Descendents of Housing

```
 Independencies : Housing
 (Housing ⟂ Risk, Saving accounts, Duration, Job, Credit History | Gender,
Credit amount, Age)
 Descendants : Housing
 []
```

Independence/Descendents of Job

```
 Independencies : Job
 (Job ⟂ Risk, Saving accounts, Age, Credit History, Gender)
 Descendants : Job
 ['Credit amount']
```

Independence/Descendents of Credit amount

```
 Independencies : Credit amount
 (Credit amount ⟂ Risk, Saving accounts, Age, Credit History, Gender | Job)
 Descendants : Credit amount
 ['Housing', 'Duration']
```

Independence/Descendents of Duration

```
 Independencies : Duration
 (Duration ⟂ Risk, Saving accounts, Age, Housing, Credit History, Job, Gender |
Credit amount)
 Descendants : Duration
 []
```

Independence/Descendents of savings accounts

```
 Independencies : Saving accounts
 (Saving accounts ⟂ Housing, Credit amount, Age, Credit History, Duration, Job,
Gender | Risk)
 Descendants : Saving accounts
 []
```

Independence/Descendents of Gender

```
 Independencies : Gender
 (Gender ⍰ Risk, Saving accounts, Credit amount, Age, Duration, Job,
Credit History)
 Descendants : Gender
 ['Housing']
```

9. Markov Blanket

Markov Blanket returns the set of nodes parent ,its children and childrens other parents
Here we try to return the markov blanket for the random variables

```
for i in model11.nodes():
  print(" markov Blanket :",i,'\n', model11.get_markov_blanket(i))
  print('\n')
```

```
markov Blanket : Risk
['Saving accounts', 'Credit History']
```

```
 markov Blanket : Credit History
 ['Risk']
```

```
 markov Blanket : Age
 ['Gender', 'Housing', 'Credit amount']
```

```
markov Blanket : Housing
['Gender', 'Credit amount', 'Age']
```

```
markov Blanket : Job
['Credit amount']
```

```
markov Blanket : Credit amount
 ['Age', 'Duration', 'Job', 'Gender', 'Housing']
```

```
markov Blanket : Duration
['Credit amount']
```

```
markov Blanket : Saving accounts
['Risk']
```

```
markov Blanket : Gender
['Housing', 'Credit amount', 'Age']
```

10. Active Trails

11.Variable Elimination using Exact Inference

Inference is same as asking conditional probability questions to the models.
There are two main categories for inference algorithms:
1. Exact Inference: These algorithms find the exact probability values for our queries.
2. Approximate Inference: These algorithms try to find approximate values by saving on computation.
We decided to use here exact inference .
There are multiple algorithms for doing exact inference. We will mainly be talking about two very common algorithms in this notebook:
1. Variable Elimination
2. Clique Tree Belief Propagation
The basic concept of variable elimination is same as doing marginalization over Joint Distribution. But variable elimination avoids computing the Joint Distribution by doing marginalization over much smaller factors. So basically if we want to eliminate $ X $ from our distribution, then we compute the product of all the factors involving $ X $ and

marginalize over them, thus allowing us to work on much smaller factors. Let’s take the student example to make things more clear:

$$P(D) = \sum_I \sum_S \sum_G \sum_L P(D, I, S, G, L)$$

$$P(D) = \sum_I \sum_S \sum_G \sum_L P(D) * P(I) * P(S|I) * P(G|D, I) * P(L|G)$$

$$P(D) = P(D) \sum_S P(S|I) \sum_I P(I) \sum_G P(G|D, I) \sum_L P(L|G)$$

In the above equation we can see that we pushed the summation inside and operated the summation only factors that involved that variable and hence avoiding computing the complete joint distribution.In pgmpy it is possible to perform inference on a Bayesian network through the Variable Elimination method

```
infer=VariableElimination(model11)
```

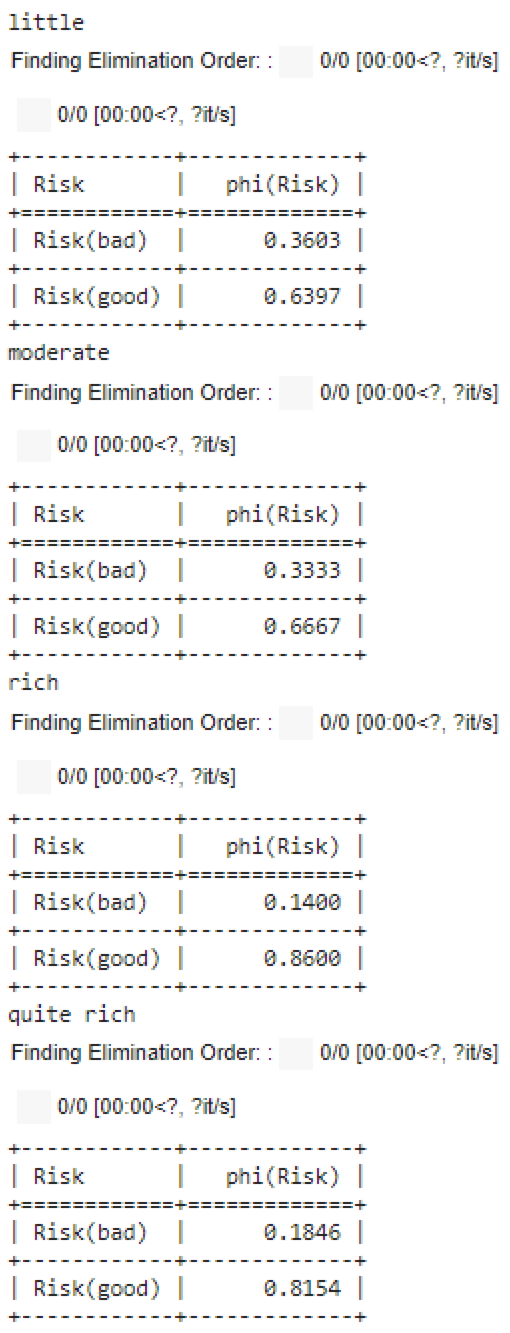
Inference1:

Given the knowledge of credit history of the borrower, what can we infer about the risk associated with the profile.Improved credit history signals better credit worthiness

```
for i in [0,1,2,3,4]:
    print("Credit History:", i)
    print(infer.query(variables=[target], evidence = {'Credit History': i}))
```

Inference2

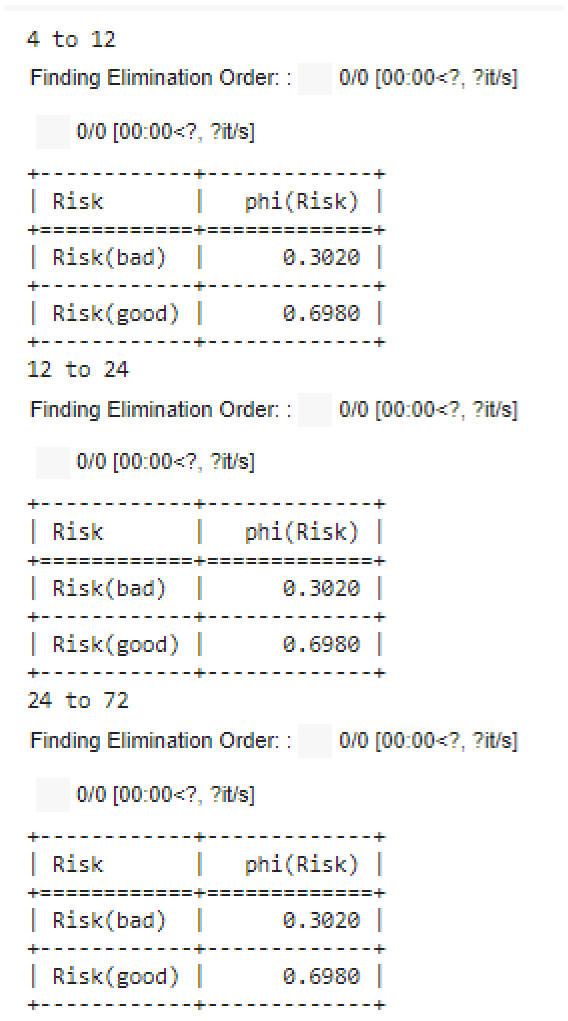
We have observed when the Saving Accounts of a person is rich and quite rich bad risk reduces and the probability of good risks are 86% and 81% in both the respective cases



```
little
Finding Elimination Order: :    0/0 [00:00<?, ?it/s]

 0/0 [00:00<?, ?it/s]

+------------+-------------+
| Risk       |   phi(Risk) |
+============+=============+
| Risk(bad)  |      0.3603 |
+------------+-------------+
| Risk(good) |      0.6397 |
+------------+-------------+
moderate
Finding Elimination Order: :    0/0 [00:00<?, ?it/s]

 0/0 [00:00<?, ?it/s]

+------------+-------------+
| Risk       |   phi(Risk) |
+============+=============+
| Risk(bad)  |      0.3333 |
+------------+-------------+
| Risk(good) |      0.6667 |
+------------+-------------+
rich
Finding Elimination Order: :    0/0 [00:00<?, ?it/s]

 0/0 [00:00<?, ?it/s]

+------------+-------------+
| Risk       |   phi(Risk) |
+============+=============+
| Risk(bad)  |      0.1400 |
+------------+-------------+
| Risk(good) |      0.8600 |
+------------+-------------+
quite rich
Finding Elimination Order: :    0/0 [00:00<?, ?it/s]

 0/0 [00:00<?, ?it/s]

+------------+-------------+
| Risk       |   phi(Risk) |
+============+=============+
| Risk(bad)  |      0.1846 |
+------------+-------------+
| Risk(good) |      0.8154 |
+------------+-------------+
```

Inference 3

Duration Loans with shorter duration are mostly repaid without default, the bad risk i.e. credit worthiness deteriorates as duration increases. Note that it justifies the intuition that the longer the period, it induces more volatility into the system, where volatility might be construed as more risk



```
4 to 12
Finding Elimination Order: :     0/0 [00:00<?, ?it/s]

   0/0 [00:00<?, ?it/s]

+------------+-------------+
| Risk       |   phi(Risk) |
+============+=============+
| Risk(bad)  |      0.3020 |
+------------+-------------+
| Risk(good) |      0.6980 |
+------------+-------------+
12 to 24
Finding Elimination Order: :     0/0 [00:00<?, ?it/s]

   0/0 [00:00<?, ?it/s]

+------------+-------------+
| Risk       |   phi(Risk) |
+============+=============+
| Risk(bad)  |      0.3020 |
+------------+-------------+
| Risk(good) |      0.6980 |
+------------+-------------+
24 to 72
Finding Elimination Order: :     0/0 [00:00<?, ?it/s]

   0/0 [00:00<?, ?it/s]

+------------+-------------+
| Risk       |   phi(Risk) |
+============+=============+
| Risk(bad)  |      0.3020 |
+------------+-------------+
| Risk(good) |      0.6980 |
+------------+-------------+
```

Inference 4

Credit amount Bad risk is maximum for high amount of credits borrowed

```
250 to 1554
```

```
Finding Elimination Order: :      0/0 [00:00<?, ?it/s]

      0/0 [00:00<?, ?it/s]
```

```
+------------+-------------+
| Risk       |   phi(Risk) |
+============+=============+
| Risk(bad)  |      0.3020 |
+------------+-------------+
| Risk(good) |      0.6980 |
+------------+-------------+
1554 to 3368
```

```
Finding Elimination Order: :      0/0 [00:00<?, ?it/s]

      0/0 [00:00<?, ?it/s]
```

```
+------------+-------------+
| Risk       |   phi(Risk) |
+============+=============+
| Risk(bad)  |      0.3020 |
+------------+-------------+
| Risk(good) |      0.6980 |
+------------+-------------+
3368 to 18424
```

```
Finding Elimination Order: :      0/0 [00:00<?, ?it/s]

      0/0 [00:00<?, ?it/s]
```

```
+------------+-------------+
| Risk       |   phi(Risk) |
+============+=============+
| Risk(bad)  |      0.3020 |
+------------+-------------+
| Risk(good) |      0.6980 |
+------------+-------------+
Credit History: 0
Finding Elimination Order: :    0/0 [00:00<?, ?it/s]

   0/0 [00:00<?, ?it/s]

+------------+-------------+
| Risk       |   phi(Risk) |
+============+=============+
| Risk(bad)  |      0.6190 |
+------------+-------------+
| Risk(good) |      0.3810 |
+------------+-------------+
Credit History: 1
Finding Elimination Order: :    0/0 [00:00<?, ?it/s]

   0/0 [00:00<?, ?it/s]

+------------+-------------+
| Risk       |   phi(Risk) |
+============+=============+
| Risk(bad)  |      0.5686 |
+------------+-------------+
| Risk(good) |      0.4314 |
+------------+-------------+
Credit History: 2
Finding Elimination Order: :    0/0 [00:00<?, ?it/s]

   0/0 [00:00<?, ?it/s]

+------------+-------------+
| Risk       |   phi(Risk) |
+============+=============+
| Risk(bad)  |      0.3195 |
+------------+-------------+
| Risk(good) |      0.6805 |
+------------+-------------+
Credit History: 3
Finding Elimination Order: :    0/0 [00:00<?, ?it/s]

   0/0 [00:00<?, ?it/s]

+------------+-------------+
| Risk       |   phi(Risk) |
+============+=============+
| Risk(bad)  |      0.3222 |
+------------+-------------+
| Risk(good) |      0.6778 |
+------------+-------------+
Credit History: 4
Finding Elimination Order: :    0/0 [00:00<?, ?it/s]

   0/0 [00:00<?, ?it/s]

+------------+-------------+
| Risk       |   phi(Risk) |
+============+=============+
| Risk(bad)  |      0.1729 |
+------------+-------------+
| Risk(good) |      0.8271 |
+------------+-------------+
```

12. Reference

Kaggle DataSet - https://www.kaggle.com/priyayadav1/german-credit/
Pgmpy Documentation - https://pgmpy.org/