```
In [59]: import os
         import numpy as np
         import pandas as pd
         from numpy.random import seed
         import tensorflow as tf
         import itertools
         from tensorflow.keras.models import Sequential

         from tensorflow.keras.layers import Dense, Activation, Dropout
         from sklearn.metrics import confusion_matrix
         import matplotlib.pyplot as plt
         from numpy import loadtxt
```

Loading dataset credicard spam detection

```
In [60]: dataset = loadtxt('Copy of creditcard.csv', delimiter=',')
         X = dataset[:,0:30]
         y = dataset[:,30]
```

Spliting the data intp train and test

```
In [62]: from sklearn.model_selection import train_test_split
         X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 20 ,
         test_size =0.2)
```

```
In [63]: X_train.shape

Out[63]: (227845, 30)
```

Standardizing data means making mean = 0 and variance = 1

```
In [64]: from sklearn.preprocessing import StandardScaler
         sc = StandardScaler().fit(X_train)
```

```
In [65]: x_train_std = sc.transform(X_train)
```

```
In [67]: x_test_std = sc.transform(X_test)
```

Sequential neural network

In [68]:

```
model_1 = Sequential([
    Dense(units=4, input_dim=30, activation='relu', name='m1_hidden1'),
    Dense(units=2, activation='relu', name='m1_hidden2'),
    Dense(units=1, activation='relu', name='m1_hidden3'),
])
```

In [69]:

```
model_1.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
m1_hidden1 (Dense)           (None, 4)                 124       
_________________________________________________________________
m1_hidden2 (Dense)           (None, 2)                 10        
_________________________________________________________________
m1_hidden3 (Dense)           (None, 1)                 3         
=================================================================
Total params: 137
Trainable params: 137
Non-trainable params: 0
_________________________________________________________________
```

In [70]:

```
model_1.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

In [71]:

```
model_1.fit(x_train_std, y_train, epochs=1, batch_size=10)
```

```
22785/22785 [==============================] - 25s 1ms/step - loss: 0.0926 - accuracy: 0.9879
```

Out[71]:

```
<tensorflow.python.keras.callbacks.History at 0x7f316ada4ef0>
```

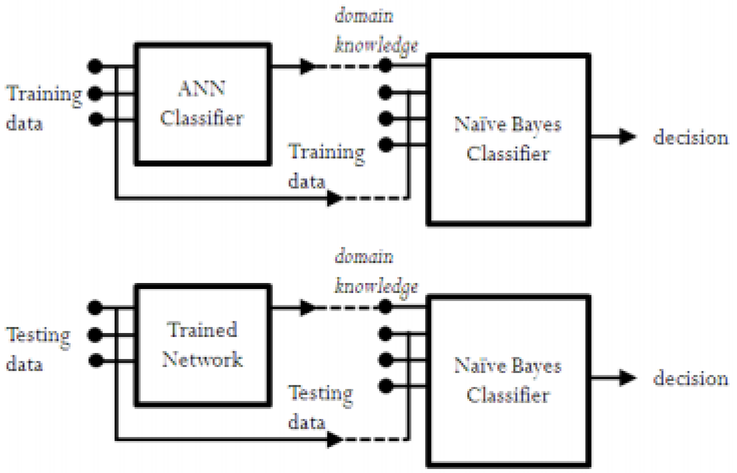
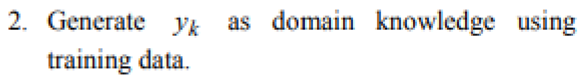


Figure 3. Integration of ANN knowledge into the Naive Bayes classifier


2. Generate y_k as domain knowledge using training data.

In [72]:
```
y_proj_train = model_1.predict(x_train_std) # making domain knowledge vaiable using training data
y_proj_test = model_1.predict(x_test_std) # predict x_test
print(np.shape(y_proj_train))
print(np.shape(y_proj_test))
```

```
(227845, 1)
(56962, 1)
```

In [73]:
```
model_1_json = model_1.to_json() # saving the above model and final weights
json_file = open("model_1.json", "w")
json_file.write(model_1_json)
json_file.close()
model_1.save_weights("model_1.h5")
print("model_1 saved to disk")
```

```
model_1 saved to disk
```

from neural netwok got accuarcy of 98.79%

In [ ]:
```
```

In [74]:
```
# linear model
from sklearn.naive_bayes import GaussianNB
model_linear = GaussianNB()
model_linear.fit(x_train_std, y_train)
```

Out[74]:
```
GaussianNB(priors=None, var_smoothing=1e-09)
```

In [75]:
```
y_pred = model_linear.predict(x_test_std)
```

In [76]:
```
from sklearn.metrics import accuracy_score
```

In [77]:
```
print("accuracy:", accuracy_score(y_test,y_pred), "\n")
```

```
accuracy: 0.9785822127032057
```

by using navie bayes classifer got accuracy of 97.85%

Combination of neural network and navie bayes classifier

In [79]:
```
model = GaussianNB()

model.fit(y_proj_train, y_train) # training navie bayes classifier with y_train and the output of neural network
y_pred = model.predict(y_proj_test)
```

In [80]:
```
print("accuracy", accuracy_score(y_test, y_pred), "\n")
```

```
accuracy 0.9994557775359011
```

by the combination of both neural network and navie bayes classifer we got accuracy of 99.94%

In [34]:
```

```

Out[34]:
```
1
```