In [ ]:
```
import pandas as pd
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import tensorflow as tf
import keras as kr
```

In [ ]:
```
loan_file = r"C:\Users\hii\Desktop\data science\loan_data1.csv"
loan_data1_df= pd.read_csv(loan_file)
```

In [ ]:
```
loan_data1_df.info()
```

In [ ]:
```
loan_data1_df.describe
```

In [ ]:
```
loan_data1_df.shape
```

In [ ]:
```
loan_data1_df.isnull().sum()
```

In [ ]:
```
loan_data1_df.columns.value_counts
```

In [ ]:
```
loan_data1_df.dtypes
```

In [ ]:
```
loan_data1_df = loan_data1_df.drop('purpose',axis=1)
loan_data1_df.head()
```

In [ ]:
```
X=loan_data1_df.drop('not.fully.paid',axis=1)
y=loan_data1_df['not.fully.paid']
```

In [ ]:
```
from sklearn.ensemble import GradientBoostingClassifier,RandomForestClassifier
gb = GradientBoostingClassifier()
rf = RandomForestClassifier()
rf.fit(X,y)
gb.fit(X,y)
print(gb.feature_importances_)
print(rf.feature_importances_)
```

In [ ]:
```
from matplotlib import pyplot as plt
import seaborn as sns
corelation=loan_data1_df.corr()
sns.heatmap(corelation, xticklabels=corelation.columns, yticklabels=corelation.colu
```

In [ ]:
```
sns.pairplot(loan_data1_df,diag_kind='hist',hue="not.fully.paid")
```

In [ ]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20,random_st
```

In [ ]:
```
#Initialize Sequential model
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Reshape((12,),input_shape=(12,)))
#Normalize the data
model.add(tf.keras.layers.BatchNormalization())
```

In [ ]:
```
#Add 1st hidden layer
model.add(tf.keras.layers.Dense(2000))
```

```
model.add(tf.keras.layers.LeakyReLU())
model.add(tf.keras.layers.BatchNormalization())
```

In [ ]:
```
#Add 2nd hidden layer
model.add(tf.keras.layers.Dense(1000))
model.add(tf.keras.layers.LeakyReLU())
model.add(tf.keras.layers.BatchNormalization())
```

In [ ]:
```
#Add 3rd hidden layer
model.add(tf.keras.layers.Dense(600))
model.add(tf.keras.layers.BatchNormalization())
model.add(tf.keras.layers.LeakyReLU())
```

In [ ]:
```
#Add 4th hidden layer
model.add(tf.keras.layers.Dense(300))
model.add(tf.keras.layers.LeakyReLU())
model.add(tf.keras.layers.BatchNormalization())
```

In [ ]:
```
#Add OUTPUT layer
model.add(tf.keras.layers.Dense(1, activation='sigmoid'))
```

In [ ]:
```
#Create optimizer with non-default learning rate
sgd_op = tf.keras.optimizers.SGD(lr=0.1, momentum=0.9, nesterov=True)

#Compile the model
model.compile(optimizer=sgd_op, loss='binary_crossentropy', metrics=['accuracy'])
```

In [ ]:
```
model.fit(X_train,y_train,
          validation_data=(X_test,y_test),
          epochs=20,
          batch_size=32)
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: