# Lending Club Loan Data Analysis

In [88]:
```
# Import required Libraries

import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import tensorflow as tf
```

In [89]:
```
# Load the data
data = pd.read_csv('/Users/manojghimire/Desktop/AIML/Assignments/my_assignment/Deep_Learning/loan_data.csv')
```

# Exploratory Data Analysis

In [90]:
```
# View and explore the dataset
data.head()
```

Out[90]:

| | credit.policy | purpose | int.rate | installment | log.annual.inc | dti | fico | days.with.cr.line | revol.bal | revol.util | inq.last.6mths | delinq.2yrs | pu |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | debt_consolidation | 0.1189 | 829.10 | 11.350407 | 19.48 | 737 | 5639.958333 | 28854 | 52.1 | 0 | 0 | |
| **1** | 1 | credit_card | 0.1071 | 228.22 | 11.082143 | 14.29 | 707 | 2760.000000 | 33623 | 76.7 | 0 | 0 | |
| **2** | 1 | debt_consolidation | 0.1357 | 366.86 | 10.373491 | 11.63 | 682 | 4710.000000 | 3511 | 25.6 | 1 | 0 | |
| **3** | 1 | debt_consolidation | 0.1008 | 162.34 | 11.350407 | 8.10 | 712 | 2699.958333 | 33667 | 73.2 | 1 | 0 | |
| **4** | 1 | credit_card | 0.1426 | 102.92 | 11.299732 | 14.97 | 667 | 4066.000000 | 4740 | 39.5 | 0 | 1 | |

In [91]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9578 entries, 0 to 9577
Data columns (total 14 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   credit.policy      9578 non-null   int64  
 1   purpose            9578 non-null   object 
 2   int.rate           9578 non-null   float64
 3   installment        9578 non-null   float64
 4   log.annual.inc     9578 non-null   float64
 5   dti                9578 non-null   float64
 6   fico               9578 non-null   int64  
 7   days.with.cr.line  9578 non-null   float64
 8   revol.bal          9578 non-null   int64  
 9   revol.util         9578 non-null   float64
 10  inq.last.6mths     9578 non-null   int64  
 11  delinq.2yrs        9578 non-null   int64  
 12  pub.rec            9578 non-null   int64  
 13  not.fully.paid     9578 non-null   int64  
dtypes: float64(6), int64(7), object(1)
memory usage: 1.0+ MB
```

In [92]:
```
# Observation:
# From the above, it is understood that all the columns have data
```

# Feature Transformation

In [93]:
```
# Check for unique values in the column 'purpose'

data.purpose.unique()
```

Out[93]:
```
array(['debt_consolidation', 'credit_card', 'all_other',
       'home_improvement', 'small_business', 'major_purchase',
       'educational'], dtype=object)
```

In [94]:

```
# The column 'purpose' is categorical. Converting the data into numeric
# Label Encoding ---> Create Dummy Variables for Column 'Purpose'

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
data['purpose'] = le.fit_transform(data['purpose'])
data
```

Out[94]:

| | credit.policy | purpose | int.rate | installment | log.annual.inc | dti | fico | days.with.cr.line | revol.bal | revol.util | inq.last.6mths | delinq.2yrs | pub.rec |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 2 | 0.1189 | 829.10 | 11.350407 | 19.48 | 737 | 5639.958333 | 28854 | 52.1 | 0 | 0 | 0 |
| **1** | 1 | 1 | 0.1071 | 228.22 | 11.082143 | 14.29 | 707 | 2760.000000 | 33623 | 76.7 | 0 | 0 | 0 |
| **2** | 1 | 2 | 0.1357 | 366.86 | 10.373491 | 11.63 | 682 | 4710.000000 | 3511 | 25.6 | 1 | 0 | 0 |
| **3** | 1 | 2 | 0.1008 | 162.34 | 11.350407 | 8.10 | 712 | 2699.958333 | 33667 | 73.2 | 1 | 0 | 0 |
| **4** | 1 | 1 | 0.1426 | 102.92 | 11.299732 | 14.97 | 667 | 4066.000000 | 4740 | 39.5 | 0 | 1 | 0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **9573** | 0 | 0 | 0.1461 | 344.76 | 12.180755 | 10.39 | 672 | 10474.000000 | 215372 | 82.1 | 2 | 0 | 0 |
| **9574** | 0 | 0 | 0.1253 | 257.70 | 11.141862 | 0.21 | 722 | 4380.000000 | 184 | 1.1 | 5 | 0 | 0 |
| **9575** | 0 | 2 | 0.1071 | 97.81 | 10.596635 | 13.09 | 687 | 3450.041667 | 10036 | 82.9 | 8 | 0 | 0 |
| **9576** | 0 | 4 | 0.1600 | 351.58 | 10.819778 | 19.18 | 692 | 1800.000000 | 0 | 3.2 | 5 | 0 | 0 |
| **9577** | 0 | 2 | 0.1392 | 853.43 | 11.264464 | 16.28 | 732 | 4740.000000 | 37879 | 57.0 | 6 | 0 | 0 |

9578 rows × 14 columns

In [95]:
```
# Check for duplicate values

df = data.drop_duplicates()
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 9578 entries, 0 to 9577
Data columns (total 14 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   credit.policy      9578 non-null   int64  
 1   purpose            9578 non-null   int64  
 2   int.rate           9578 non-null   float64
 3   installment        9578 non-null   float64
 4   log.annual.inc     9578 non-null   float64
 5   dti                9578 non-null   float64
 6   fico               9578 non-null   int64  
 7   days.with.cr.line  9578 non-null   float64
 8   revol.bal          9578 non-null   int64  
 9   revol.util         9578 non-null   float64
 10  inq.last.6mths     9578 non-null   int64  
 11  delinq.2yrs        9578 non-null   int64  
 12  pub.rec            9578 non-null   int64  
 13  not.fully.paid     9578 non-null   int64  
dtypes: float64(6), int64(8)
memory usage: 1.1 MB
```

In [96]:
```
# Check for Null values, if any

df.isnull().any()
```

Out[96]:
```
credit.policy        False
purpose              False
int.rate             False
installment          False
log.annual.inc       False
dti                  False
fico                 False
days.with.cr.line    False
revol.bal            False
revol.util           False
inq.last.6mths       False
delinq.2yrs          False
pub.rec              False
not.fully.paid       False
dtype: bool
```

In [97]:
```
# Check whether the dataset is balanced or unbalanced

df['credit.policy'].value_counts()
```

Out[97]:
```
1    7710
0    1868
Name: credit.policy, dtype: int64
```

In [98]:
```
# The dataset looks to be unbalanced
```

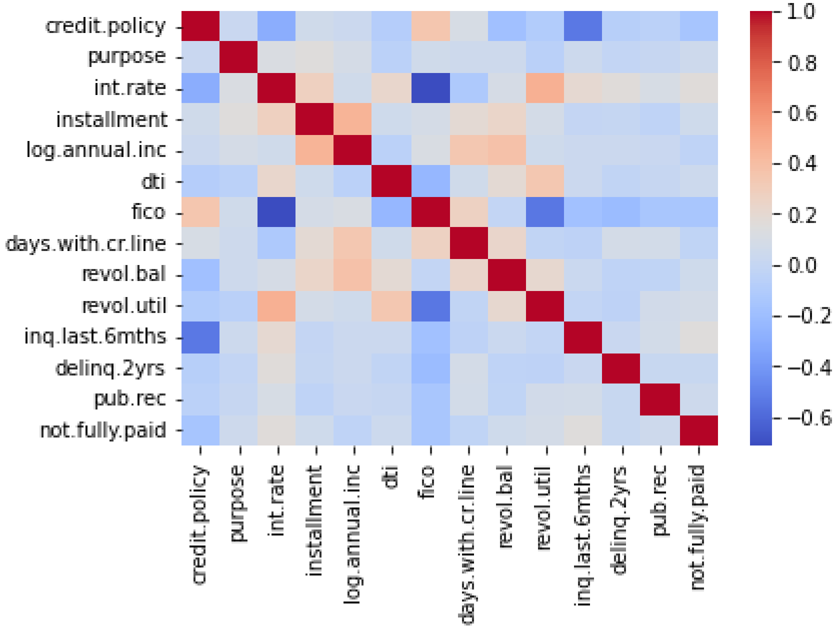
# Feature Engineering

In [99]:
```
# Check the descriptive statistics for the given dataset

df.describe()
```

Out[99]:

| | credit.policy | purpose | int.rate | installment | log.annual.inc | dti | fico | days.with.cr.line | revol.bal | revol.util | in |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **count** | 9578.000000 | 9578.000000 | 9578.000000 | 9578.000000 | 9578.000000 | 9578.000000 | 9578.000000 | 9578.000000 | 9.578000e+03 | 9578.000000 | |
| **mean** | 0.804970 | 1.944038 | 0.122640 | 319.089413 | 10.932117 | 12.606679 | 710.846314 | 4560.767197 | 1.691396e+04 | 46.799236 | |
| **std** | 0.396245 | 1.686881 | 0.026847 | 207.071301 | 0.614813 | 6.883970 | 37.970537 | 2496.930377 | 3.375619e+04 | 29.014417 | |
| **min** | 0.000000 | 0.000000 | 0.060000 | 15.670000 | 7.547502 | 0.000000 | 612.000000 | 178.958333 | 0.000000e+00 | 0.000000 | |
| **25%** | 1.000000 | 1.000000 | 0.103900 | 163.770000 | 10.558414 | 7.212500 | 682.000000 | 2820.000000 | 3.187000e+03 | 22.600000 | |
| **50%** | 1.000000 | 2.000000 | 0.122100 | 268.950000 | 10.928884 | 12.665000 | 707.000000 | 4139.958333 | 8.596000e+03 | 46.300000 | |
| **75%** | 1.000000 | 2.000000 | 0.140700 | 432.762500 | 11.291293 | 17.950000 | 737.000000 | 5730.000000 | 1.824950e+04 | 70.900000 | |
| **max** | 1.000000 | 6.000000 | 0.216400 | 940.140000 | 14.528354 | 29.960000 | 827.000000 | 17639.958330 | 1.207359e+06 | 119.000000 | |

In [100]:
```
# Visualize the correlation coefficient between various variables using Heatmap

coRR = df.corr(method='pearson')
coRR
map = sns.heatmap(coRR, cmap='coolwarm')
plt.show()
```




In [101]:
```
# Observation:
# From the above Heatmap, it is concluded that all the features are important and needs to be considered
# for building the model
```

In [102]:
```
# Define features and label

features = df.iloc[:,1:]
label = df.iloc[:,[0]]
```

In [103]:
```
features
```

Out[103]:

| | purpose | int.rate | installment | log.annual.inc | dti | fico | days.with.cr.line | revol.bal | revol.util | inq.last.6mths | delinq.2yrs | pub.rec | not.fully.paid |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 2 | 0.1189 | 829.10 | 11.350407 | 19.48 | 737 | 5639.958333 | 28854 | 52.1 | 0 | 0 | 0 | 0 |
| **1** | 1 | 0.1071 | 228.22 | 11.082143 | 14.29 | 707 | 2760.000000 | 33623 | 76.7 | 0 | 0 | 0 | 0 |
| **2** | 2 | 0.1357 | 366.86 | 10.373491 | 11.63 | 682 | 4710.000000 | 3511 | 25.6 | 1 | 0 | 0 | 0 |
| **3** | 2 | 0.1008 | 162.34 | 11.350407 | 8.10 | 712 | 2699.958333 | 33667 | 73.2 | 1 | 0 | 0 | 0 |
| **4** | 1 | 0.1426 | 102.92 | 11.299732 | 14.97 | 667 | 4066.000000 | 4740 | 39.5 | 0 | 1 | 0 | 0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **9573** | 0 | 0.1461 | 344.76 | 12.180755 | 10.39 | 672 | 10474.000000 | 215372 | 82.1 | 2 | 0 | 0 | 1 |
| **9574** | 0 | 0.1253 | 257.70 | 11.141862 | 0.21 | 722 | 4380.000000 | 184 | 1.1 | 5 | 0 | 0 | 1 |
| **9575** | 2 | 0.1071 | 97.81 | 10.596635 | 13.09 | 687 | 3450.041667 | 10036 | 82.9 | 8 | 0 | 0 | 1 |
| **9576** | 4 | 0.1600 | 351.58 | 10.819778 | 19.18 | 692 | 1800.000000 | 0 | 3.2 | 5 | 0 | 0 | 1 |
| **9577** | 2 | 0.1392 | 853.43 | 11.264464 | 16.28 | 732 | 4740.000000 | 37879 | 57.0 | 6 | 0 | 0 | 1 |

9578 rows × 13 columns

In [104]:
```
label
```

Out[104]:

| | credit.policy |
|---|---|
| **0** | 1 |
| **1** | 1 |
| **2** | 1 |
| **3** | 1 |
| **4** | 1 |
| **...** | ... |
| **9573** | 0 |
| **9574** | 0 |
| **9575** | 0 |
| **9576** | 0 |
| **9577** | 0 |

9578 rows × 1 columns

In [105]:
```
# Standardization

from sklearn.preprocessing import StandardScaler
Scaler = StandardScaler()
features = Scaler.fit_transform(features)
features
```

Out[105]:
```
array([[ 0.03317632, -0.13931753,  2.46309947, ..., -0.29973008,
        -0.23700318, -0.43652393],
       [-0.55966463, -0.57886837, -0.43885443, ..., -0.29973008,
        -0.23700318, -0.43652393],
       [ 0.03317632,  0.48648368,  0.23070836, ..., -0.29973008,
        -0.23700318, -0.43652393],
       ...,
       [ 0.03317632, -0.57886837, -1.06867038, ..., -0.29973008,
        -0.23700318,  2.29082517],
       [ 1.2188582 ,  1.39166043,  0.1569135 , ..., -0.29973008,
        -0.23700318,  2.29082517],
       [ 0.03317632,  0.61685894,  2.58060136, ..., -0.29973008,
        -0.23700318,  2.29082517]])
```

# Modeling

In [106]:

```
# Recursive Feature Elimination (RFE ) Technique

# Initialize the model algorithm
from sklearn.linear_model import LinearRegression
modelLR = LinearRegression()

# Apply RFE to model (All features and labels)
from sklearn.feature_selection import RFE
selectFeaturesFromRFE = RFE(estimator=modelLR,
                            step=1)

# Fit the data with RFE
selectFeaturesFromRFE.fit(features,label)

# Get Features with High Ranking (1,2,3,4,...) (Get features that has Rank 1. Sometimes Rank 2 is considered)

print(selectFeaturesFromRFE.ranking_)
```

```
[4 1 2 1 7 1 5 1 1 1 6 8 3]

/opt/anaconda3/lib/python3.7/site-packages/sklearn/utils/validation.py:724: DataConversionWarning: A column-v
ector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example u
sing ravel().
  y = column_or_1d(y, warn=True)
```

In [107]:

```
# Observation:
# The features 'int.rate', 'log.annual.inc', 'fico', 'revol.bal', 'revol.util' and 'inq.last.6mths' are
# selected for model building
```

In [108]:
```
# Select by Model (SBM)

# 1. Initialize the model algorithm
from sklearn.linear_model import LinearRegression
modelLR = LinearRegression()

# 2. Apply SBM to model (All features and labels)
from sklearn.feature_selection import SelectFromModel
selectFeaturesFromSBM = SelectFromModel(modelLR)

# Fit the data with SBM
selectFeaturesFromSBM.fit(features,label)

# 3. Get Features with High Ranking (1,2,3,4,...) (Get features that has Rank 1. Sometimes Rank 2 is considered
print(selectFeaturesFromSBM.get_support())
```

```
[False False False False False  True False  True False  True False False
 False]
```

In [109]:
```
# Observation:
# The features 'fico', 'revol.bal' and 'inq.last.6mths' are selected for model building
```

In [110]:
```
finalFeatures=features[:,[2,4,6,8,9,10]]
```

In [111]:
```
seed = 2501

X_train,X_test,y_train,y_test = train_test_split(finalFeatures,
                                                 label,
                                                 test_size=0.2,
                                                 random_state = seed)
tf.random.set_seed(seed)
np.random.seed(seed)
```

In [112]:
```
#Architect the model
model = tf.keras.models.Sequential()

# units ---> How many neurons/nodes needs to initialized?
# Option1: No of units ---> No of features
# Option2: No of units ---> 3 * No of features (3*6=18)
# Option3: No of units ---> 1/3 * No of features

model.add(tf.keras.layers.Dense( units = 18, activation= 'relu',input_shape=(6,) ))
model.add(tf.keras.layers.Dense( units = 18, activation= 'relu' ))
model.add(tf.keras.layers.Dense( units = 18, activation= 'relu' ))
model.add(tf.keras.layers.Dense( units = 1, activation= 'sigmoid' ))
```

In [113]:
```
# Compile Model
# Binary Classification : binary_crossentropy
# MultiClass (Label) Classification: categorical_crossentropy

model.compile(optimizer = "Adam" ,
              loss = 'binary_crossentropy',
              metrics = ['accuracy'])
```

In [114]:
```
# Custom EarlyStopping Code

class thresholdCallback(tf.keras.callbacks.Callback):
    def __init__(self, cl):
        super(thresholdCallback, self).__init__()
        self.cl = cl

    def on_epoch_end(self, epoch, logs=None):
        test_score = logs["val_accuracy"]
        train_score = logs["accuracy"]
        if ( test_score > train_score and test_score > self.cl ) or test_score == 1 :
            self.model.stop_training = True
```

In [115]:
```
# Train the model

scoreMonitor = thresholdCallback(cl=0.9)

epoch_hist = model.fit(X_train,
                       y_train,
                       epochs=300,
                       validation_data=(X_test,y_test),
                       callbacks= [scoreMonitor] )
```

```
Epoch 1/300
240/240 [==============================] - 1s 3ms/step - loss: 0.4181 - accuracy: 0.8536 - val_loss: 0.3317 -
val_accuracy: 0.8800
Epoch 2/300
240/240 [==============================] - 0s 2ms/step - loss: 0.3159 - accuracy: 0.8824 - val_loss: 0.3112 -
val_accuracy: 0.8904
Epoch 3/300
240/240 [==============================] - 0s 2ms/step - loss: 0.2984 - accuracy: 0.8889 - val_loss: 0.3058 -
val_accuracy: 0.8899
Epoch 4/300
240/240 [==============================] - 0s 2ms/step - loss: 0.2897 - accuracy: 0.8910 - val_loss: 0.2971 -
val_accuracy: 0.8935
Epoch 5/300
240/240 [==============================] - 0s 2ms/step - loss: 0.2832 - accuracy: 0.8928 - val_loss: 0.2946 -
val_accuracy: 0.8920
Epoch 6/300
240/240 [==============================] - 0s 2ms/step - loss: 0.2778 - accuracy: 0.8956 - val_loss: 0.2921 -
val_accuracy: 0.8941
Epoch 7/300
240/240 [==============================] - 0s 2ms/step - loss: 0.2720 - accuracy: 0.8990 - val_loss: 0.2850 -
val_accuracy: 0.8946
Epoch 8/300
240/240 [==============================] - 0s 2ms/step - loss: 0.2683 - accuracy: 0.8983 - val_loss: 0.2833 -
val_accuracy: 0.8946
Epoch 9/300
240/240 [==============================] - 0s 2ms/step - loss: 0.2653 - accuracy: 0.9050 - val_loss: 0.2789 -
val_accuracy: 0.8987
Epoch 10/300
240/240 [==============================] - 0s 2ms/step - loss: 0.2618 - accuracy: 0.9054 - val_loss: 0.2743 -
val_accuracy: 0.8987
Epoch 11/300
```

```
240/240 [==============================] - 0s 2ms/step - loss: 0.2598 - accuracy: 0.9081 - val_loss: 0.2722 -
val_accuracy: 0.8982
Epoch 12/300
240/240 [==============================] - 0s 2ms/step - loss: 0.2563 - accuracy: 0.9103 - val_loss: 0.2700 -
val_accuracy: 0.9040
Epoch 13/300
240/240 [==============================] - 0s 2ms/step - loss: 0.2537 - accuracy: 0.9114 - val_loss: 0.2659 -
val_accuracy: 0.9045
Epoch 14/300
240/240 [==============================] - 0s 2ms/step - loss: 0.2509 - accuracy: 0.9129 - val_loss: 0.2671 -
val_accuracy: 0.9113
Epoch 15/300
240/240 [==============================] - 0s 2ms/step - loss: 0.2484 - accuracy: 0.9133 - val_loss: 0.2611 -
val_accuracy: 0.9108
Epoch 16/300
240/240 [==============================] - 0s 2ms/step - loss: 0.2462 - accuracy: 0.9156 - val_loss: 0.2592 -
val_accuracy: 0.9113
Epoch 17/300
240/240 [==============================] - 0s 2ms/step - loss: 0.2449 - accuracy: 0.9156 - val_loss: 0.2581 -
val_accuracy: 0.9128
Epoch 18/300
240/240 [==============================] - 0s 2ms/step - loss: 0.2428 - accuracy: 0.9174 - val_loss: 0.2581 -
val_accuracy: 0.9123
Epoch 19/300
240/240 [==============================] - 0s 2ms/step - loss: 0.2410 - accuracy: 0.9171 - val_loss: 0.2538 -
val_accuracy: 0.9144
Epoch 20/300
240/240 [==============================] - 1s 2ms/step - loss: 0.2390 - accuracy: 0.9203 - val_loss: 0.2527 -
val_accuracy: 0.9212
```

In [116]:
```
# Validate the model
# Since the dataset is an UNBALANCED DATASET , it is recommended to perform one more check,
# Domainwise Tolerance !!! or F1 Score

from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

# Prediction

predLabel= (model.predict(finalFeatures) > 0.5).astype("int32")
confusion_matrix(label,predLabel)
```

Out[116]:
```
array([[1287,  581],
       [ 170, 7540]])
```

In [117]:
```
print(classification_report(label,predLabel))
```

```
              precision    recall  f1-score   support

           0       0.88      0.69      0.77      1868
           1       0.93      0.98      0.95      7710

    accuracy                           0.92      9578
   macro avg       0.91      0.83      0.86      9578
weighted avg       0.92      0.92      0.92      9578
```

In [87]:
```
# Observation:
# Considering the average of (Recall of 1 and Precision of 0) (0.98+0.88)/2 = 0.93

# Since the value 0.93 > CL 0.90, the model is Accepted
```

In [ ]: