Data Sciene Project 3

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline
```

Data Cleaning and Regression

In this project, I will be loading, cleaning and transforming a small set of data related to loan applications.

Data Preparation and Exploration

In [2]:
```
df_loan = pd.read_csv('../data/loan.csv', index_col='CustomerID')
df_loan.shape
```

Out[2]: (663, 4)

In [3]:
```
# 'CustomerID' is a unique id that should be set as the index using th
e index_col argument.
# Storing this dataframe as df_borrower.
df_borrower = pd.read_csv('../data/borrower.csv', index_col='CustomerI
D')


df_borrower.shape
```

Out[3]: (663, 2)

In [4]:
```
# Joining the datasets and store as df.

df_new = pd.merge(df_loan,
                  df_borrower,
                  on='CustomerID',
                  how='inner')

# Printing information summary using 'info'
df_new.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 663 entries, 2 to 750
Data columns (total 6 columns):
WasTheLoanApproved          663 non-null object
LoanReason                  663 non-null object
LoanPayoffPeriodInMonths    663 non-null int64
RequestedAmount             663 non-null int64
Age                         585 non-null float64
YearsAtCurrentEmployer      542 non-null object
dtypes: float64(1), int64(2), object(3)
memory usage: 36.3+ KB
```

In [5]:
```
# Loan reason is a categorical variable.
# Printing the counts of each category using 'value_counts'
df_new['LoanReason'].value_counts()
```

Out[5]:
```
goods     312
auto      217
other      90
school     44
Name: LoanReason, dtype: int64
```

In [6]:
```
# Transforming LoanReason Using One-Hot Encoding


df_loanreason = pd.get_dummies(df_new.LoanReason, prefix= 'LoanReason'
)
df_loanreason.head()
```

Out[6]:

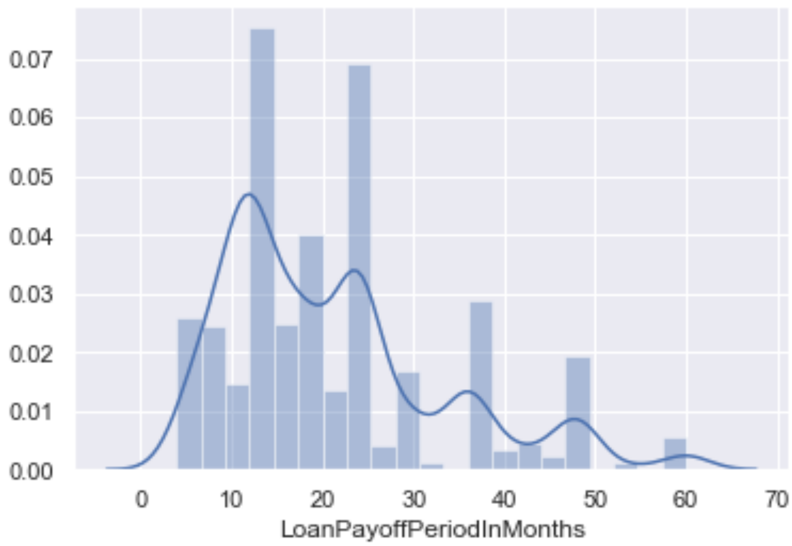
	LoanReason_auto	LoanReason_goods	LoanReason_other	LoanReason_school
CustomerID				
2	0	1	0	0
3	1	0	0	0
4	1	0	0	0
5	0	1	0	0
6	0	0	1	0

In [7]:
```
# Creating Transformed Feature Dataframe


df_features = df_loanreason.copy()

df_features.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 663 entries, 2 to 750
Data columns (total 4 columns):
LoanReason_auto      663 non-null uint8
LoanReason_goods     663 non-null uint8
LoanReason_other     663 non-null uint8
LoanReason_school    663 non-null uint8
dtypes: uint8(4)
memory usage: 7.8 KB
```

In [8]:
```
# ploting LoanPayoffPeriodInMonths using default settings.
sns.set()
sns.distplot(df_new.LoanPayoffPeriodInMonths)
```

Out[8]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a174d3090>
```



In [9]:
```
# Creating Period Bins

# Period bins is given by
#   minimum value in LoanPayoffPeriodInMonths
#   12
#   24
#   maximum value of LoanPayoffPeriodInMonths

preiod_bins= [min(df_new.LoanPayoffPeriodInMonths), 12, 24, max(df_new
.LoanPayoffPeriodInMonths)]

print(preiod_bins)
```

```
[4, 12, 24, 60]
```

In [10]:
```
# Bin LoanPayoffPeriodInMonths


# Creadin the bin labels as on the period bins ['0','1','2+'].
# Stored as loanperiod_years
df_new1 = df_new.copy()
loanperiod_years = pd.cut(df_new.LoanPayoffPeriodInMonths, bins=preiod
_bins, labels=['0','1','2+'],
                                        include_lowest=True
                                       )
loanperiod_years.value_counts()
```

Out[10]:
```
1     268
0     241
2+    154
Name: LoanPayoffPeriodInMonths, dtype: int64
```

In [11]:
```
# 10. (2pts) Transforming Period Year Bins as One-Hot Encoding


df_loanperiod = pd.get_dummies(loanperiod_years, prefix= 'LoanPeriodYe
ars')
df_loanperiod.head()
```

Out[11]:

	LoanPeriodYears_0	LoanPeriodYears_1	LoanPeriodYears_2+
CustomerID			
2	1	0	0
3	1	0	0
4	1	0	0
5	0	1	0
6	0	1	0

In [12]:
```
# Joining the existing df_features dataframe with df_loanperiod.

df_features = pd.merge(df_features,
                       df_loanperiod,
                       on='CustomerID',
                       how='inner')

df_features.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 663 entries, 2 to 750
Data columns (total 7 columns):
LoanReason_auto       663 non-null uint8
LoanReason_goods      663 non-null uint8
LoanReason_other      663 non-null uint8
LoanReason_school     663 non-null uint8
LoanPeriodYears_0     663 non-null uint8
LoanPeriodYears_1     663 non-null uint8
LoanPeriodYears_2+    663 non-null uint8
dtypes: uint8(7)
memory usage: 9.7 KB
```

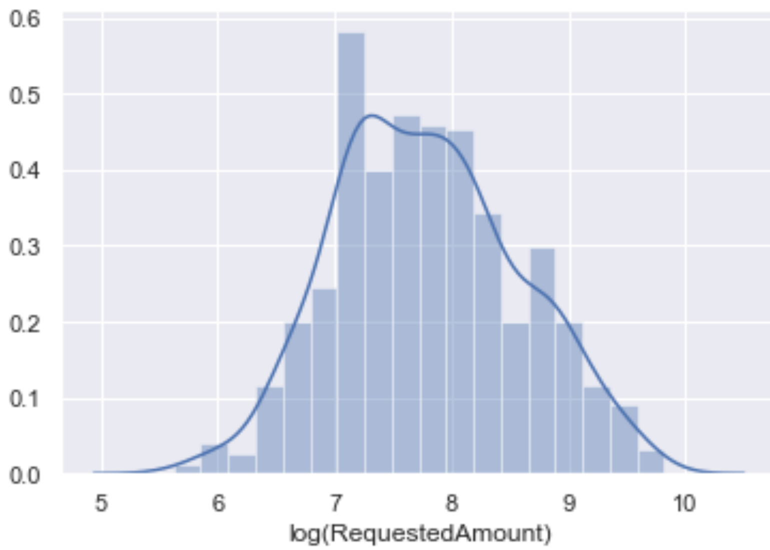
In [13]:
```
# ploting RequestedAmount using default settings.
sns.distplot(df_new.RequestedAmount)
```

Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x1a1768b6d0>

This feature is very skewed and has a very wide range.

In [14]:
```
#  Log Transforming RequestedAmount

requestedamount_log = df_new['RequestedAmount'].apply(np.log)

# ploting the transformed variable using default settings.

sns.distplot(requestedamount_log).set(xlabel='log(RequestedAmount)')
```

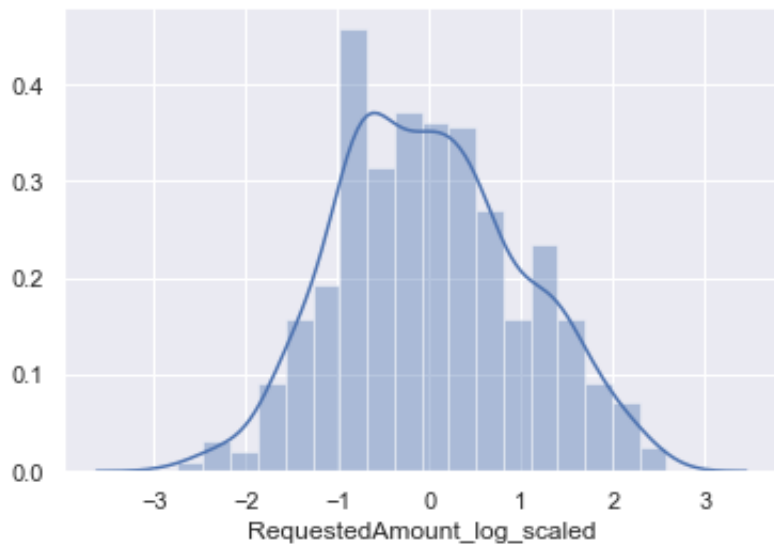
Out[14]:
```
[Text(0.5, 0, 'log(RequestedAmount)')]
```



Note that the shape is much more 'normal' now

In [15]:
```
# Centering and Scaling log(RequestedAmount) Manually


RequestedAmount_log_scaled = (requestedamount_log - np.mean(requestedamount_log))/ np.std(requestedamount_log)

# Ploting RequestedAmount_log_scaled.
sns.distplot(RequestedAmount_log_scaled).set(xlabel='RequestedAmount_log_scaled')
```

Out[15]:
```
[Text(0.5, 0, 'RequestedAmount_log_scaled')]
```



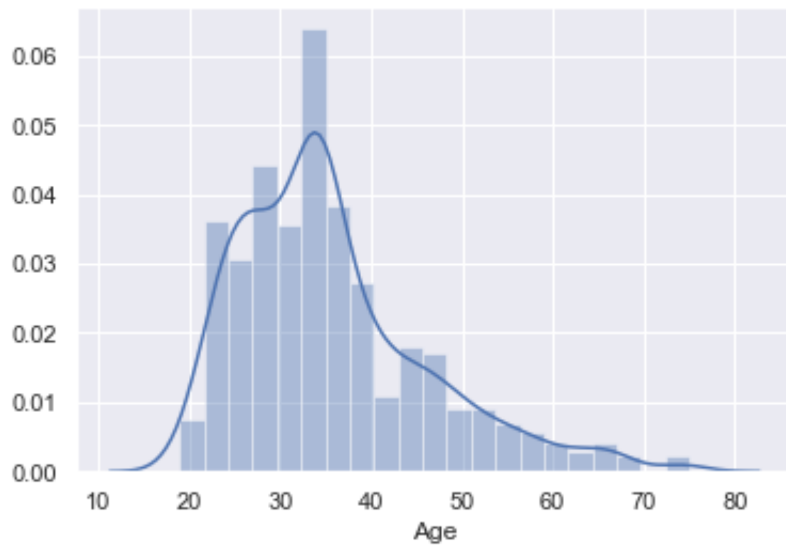
Note that data has been centered and scaled.

In [16]:
```
# The Age variable has missing values.
# Before we fill the missing values, create a dummy column noting where data is missing. We want to store this as an int instead of a boolean.
df_features = pd.merge(df_features,
                      df_new['Age'].isnull().astype(int),
                      on='CustomerID',
                      how='inner')

# Printing the number of 0s and 1s in Age_missing using 'value_counts'.
df_features.Age.value_counts()
```

Out[16]:
```
0    585
1     78
Name: Age, dtype: int64
```

In [17]:
```
# Fill Age with Median


df_new['Age'] = df_new['Age'].fillna(np.nanmedian(df_new['Age']))

# ploting Age.
sns.distplot(df_new['Age'])
```

Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0x1a179c14d0>



In [18]:
```
# Centering and Scaling Age Using StandardScaler


from sklearn.preprocessing import StandardScaler


df_features['Age_scaled'] = StandardScaler().fit_transform(df_new[['Ag
e']])

print(np.mean(df_features['Age_scaled']))
print(np.std(df_features['Age_scaled']))
```

```
2.1166242882748084e-16
1.0
```

In [19]:
```
# There are missing values in YearsAtCurrentEmployer as well.
# Since this is a categorical feature, we'll fill with the most common
value (mode).

df_new.YearsAtCurrentEmployer.value_counts(dropna=False)
```

Out[19]:
```
4      183
10+    135
NaN    121
7       98
1       97
0       29
Name: YearsAtCurrentEmployer, dtype: int64
```

In [20]:
```
# Geting Mode of YearsAtCurrentEmployer



print(years_mode)
```

```
4
```

In [21]:
```
# Filling Missing in YearsAtCurrentEmployer With Mode


df_new.YearsAtCurrentEmployer = df_new.YearsAtCurrentEmployer.fillna(y
ears_mode)

df_new.YearsAtCurrentEmployer.value_counts(dropna=False)
```

Out[21]:
```
4      304
10+    135
7       98
1       97
0       29
Name: YearsAtCurrentEmployer, dtype: int64
```

In [22]:
```
# One-Hot Encode YearsAtCurrentEmployer

df_employed = pd.get_dummies(df_new.YearsAtCurrentEmployer, prefix= 'Y
earsAtCurrentEmployer')

# Printing 'head' of df_employed to confirm the transformation.
df_employed.head()
```

Out[22]:

	YearsAtCurrentEmployer_0	YearsAtCurrentEmployer_1	YearsAtCurrentEmployer_10+
CustomerID			
2	0	0	0
3	0	0	0
4	0	0	1
5	0	0	0
6	0	0	1

In [23]:
```
# Extend Transformed Features with YearsAtCurrentEmployer


df_features = pd.merge(df_features,
                      df_employed,
                      on='CustomerID',
                      how='inner')

# Printing df_features information summary using 'info'.

df_features.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 663 entries, 2 to 750
Data columns (total 14 columns):
LoanReason_auto              663 non-null uint8
LoanReason_goods             663 non-null uint8
LoanReason_other             663 non-null uint8
LoanReason_school            663 non-null uint8
LoanPeriodYears_0            663 non-null uint8
LoanPeriodYears_1            663 non-null uint8
LoanPeriodYears_2+           663 non-null uint8
Age                          663 non-null int64
Age_scaled                   663 non-null float64
YearsAtCurrentEmployer_0     663 non-null uint8
YearsAtCurrentEmployer_1     663 non-null uint8
YearsAtCurrentEmployer_10+   663 non-null uint8
YearsAtCurrentEmployer_4     663 non-null uint8
YearsAtCurrentEmployer_7     663 non-null uint8
dtypes: float64(1), int64(1), uint8(12)
memory usage: 23.3 KB
```

PCA and K-Means

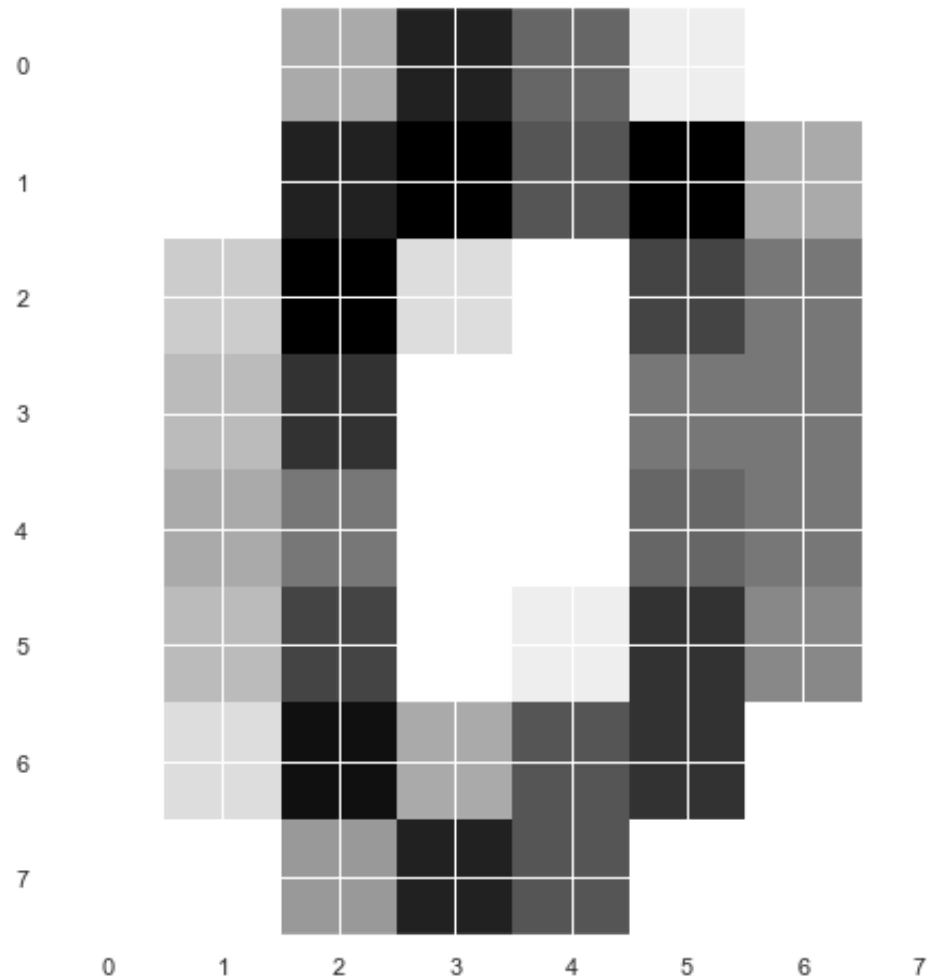
The MNIST digits dataset is composed of a set of images of handwritten digits from 0 to 9. There are 1797 images, each 8x8 pixels. If we flatten out each image we get a dataset of 1797 observations, each with 64 features, each belonging to one of 10 classes. Here we'll reduce dimensionality to 2-D to see if the data clusters by class. This a a typical data science practice problem that I did.

In [24]:
```
# Load the Digits Dataset



from sklearn.datasets import load_digits

# Loading the dataset into 'digits' using load_digits
digits = load_digits()

# Extracting digits['data'] to X_digits. No need to reshape.
X_digits = digits['data']

# Extracting the labels in digits['target'] to y_digits
y_digits = digits['target']

# Printing the shape of X_digits (should be 1797 rows, 64 columns).
X_digits.shape
```

Out[24]:
```
(1797, 64)
```

In [25]:
```
#Lets see what the image looks like
fig, ax = plt.subplots(figsize=(8, 8))
ax.imshow(digits['images'][0], cmap=plt.cm.gray_r)
```

Out[25]:
```
<matplotlib.image.AxesImage at 0x1a18233a10>
```



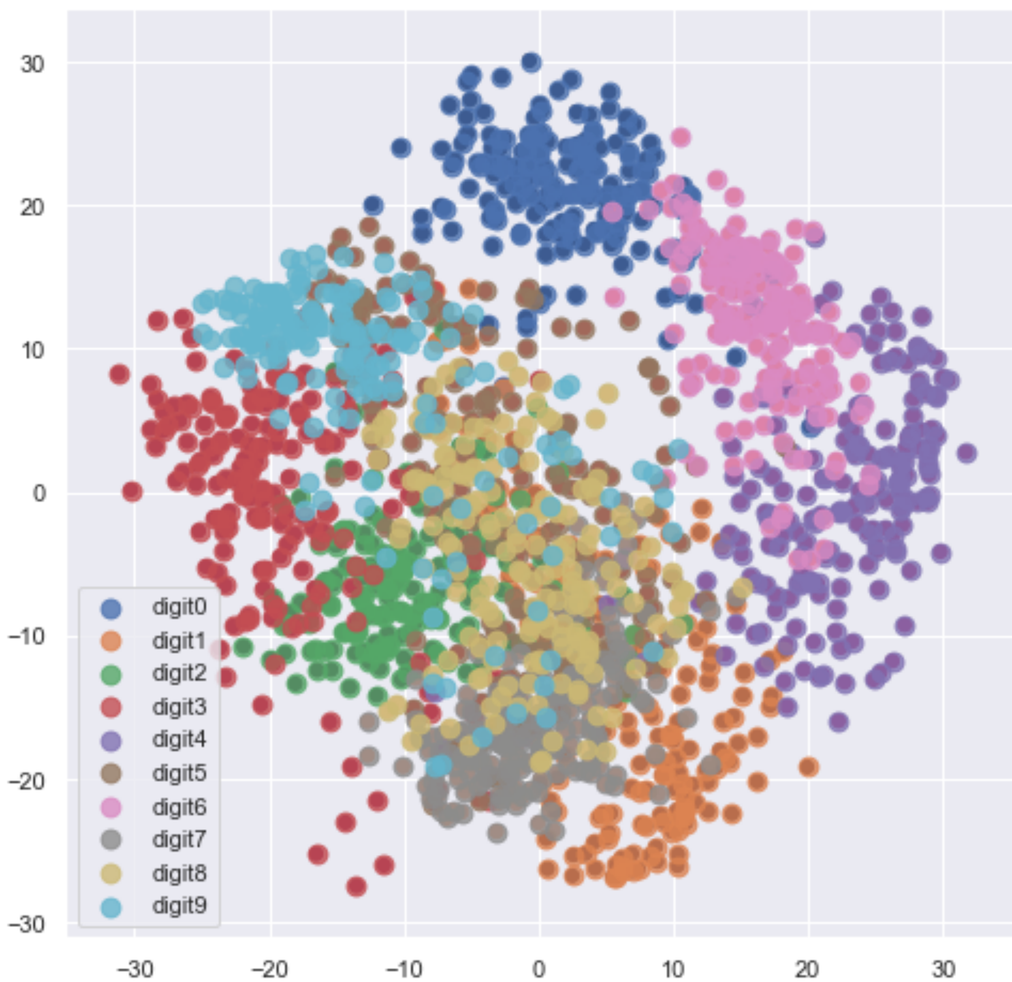
In [26]:
```
# Import PCA from sklearn
from sklearn.decomposition import PCA

pca = PCA(n_components=2, random_state=123)
```

In [27]:
```
# Transforming X_digits Using PCA


X_2D = pca.fit_transform(X_digits)

# Shape of X_2D
X_2D.shape
```

Out[27]: (1797, 2)

In [28]:
```
# Ploting PCA Representation Colored by Labels


# Creating a single figure and axis of size 8,8 using plt.subplots.
fig, ax = plt.subplots(figsize=(8, 8))
ax.scatter(X_2D[:, 0], X_2D[:, 1],
            c=y_digits)


for category in range(10):

    X_subset = X_2D[y_digits == category]

    ax.scatter(X_subset[:, 0], X_subset[:, 1], s= 80, alpha = 0.8, lab
el = 'digit' + str(category))

ax.legend()
```

Out[28]: <matplotlib.legend.Legend at 0x1a18071d90>



K-Means Clustering

How clustered are our classes? Can k-Means find clusters in the 2D PCA transformed data that at all correspond to the plot seen above?

In [29]:
```
from sklearn.cluster import KMeans

km = KMeans(n_clusters= 10, random_state=123)
```

In [30]:
```
# Generating Cluster Assignments

cluster_assignments = km.fit_predict(X_2D)

# Note: cluster assignment values will be from 0 to 9
print(cluster_assignments[0:10])
```

```
[5 0 9 4 7 8 7 3 1 1]
```

In [31]:
```
# Plotting PCA Representation Colored by Cluster Assignment

fig, ax = plt.subplots(figsize=(8, 8))




for cluster in range(10):
    X_subset = X_2D[cluster_assignments == cluster]
    ax.scatter(X_subset[:, 0], X_subset[:, 1], s=80, alpha = 0.8, labe
l = 'cluster ' + str(cluster))
    for i in range(len(km.cluster_centers_)):
      ax.scatter(km.cluster_centers_[:, 0], km.cluster_centers_[:, 1],
marker='x', c='k', label=None)

# Add a legend to the plot.
ax.legend()
```

Out[31]: <matplotlib.legend.Legend at 0x1a181c2850>