- Importing libraries

In [11]:

```
import itertools
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.ticker import NullFormatter
import pandas as pd
import numpy as np
import matplotlib.ticker as ticker
from sklearn import preprocessing
%matplotlib inline
import seaborn as sns
```

- Downloading the dataset

In [2]:

```
!wget -O loan_train.csv https://s3-api.us-geo.objectstorage.softlayer.net/c
```

```
zsh:1: command not found: wget
```

- Reading the CSV file

In [3]:

```
df = pd.read_csv('loan_train.csv')
```

- Showing the first five rows

In [4]:

```
df.head()
```

Out[4]:

	Unnamed: 0	Unnamed: 0.1	loan_status	Principal	terms	effective_date	due_date	age	edu
0	0	0	PAIDOFF	1000	30	9/8/2016	10/7/2016	45	Sc
1	2	2	PAIDOFF	1000	30	9/8/2016	10/7/2016	33	B
2	3	3	PAIDOFF	1000	15	9/8/2016	9/22/2016	27	
3	4	4	PAIDOFF	1000	30	9/9/2016	10/8/2016	28	
4	6	6	PAIDOFF	1000	30	9/9/2016	10/8/2016	29	

- Shape of the data

In [5]:

```
df.shape
```

Out[5]:

```
(346, 10)
```

- Dropping columns Unnamed: 0 and Unnamed: 1

In [7]:

```
df =df.drop(['Unnamed: 0','Unnamed: 0.1'],axis=1)
```

In [8]:

```
df.head()
```

Out[8]:

	loan_status	Principal	terms	effective_date	due_date	age	education	Gender
0	PAIDOFF	1000	30	9/8/2016	10/7/2016	45	High School or	male

	loan_status	Principal	terms	effective_date	due_date	age	education	Gender
							Below	
1	PAIDOFF	1000	30	9/8/2016	10/7/2016	33	Bechalor	female
2	PAIDOFF	1000	15	9/8/2016	9/22/2016	27	college	male
3	PAIDOFF	1000	30	9/9/2016	10/8/2016	28	college	female

- COnverting columns effective_date and due_date to Pandas datetime

In [9]:

```
df['effective_date'] = pd.to_datetime(df['effective_date'])
df['due_date'] = pd.to_datetime(df['due_date'])
```

In [10]:

```
df.head()
```

Out[10]:

	loan_status	Principal	terms	effective_date	due_date	age	education	Gender
0	PAIDOFF	1000	30	2016-09-08	2016-10-07	45	High School or Below	male
1	PAIDOFF	1000	30	2016-09-08	2016-10-07	33	Bechalor	female
2	PAIDOFF	1000	15	2016-09-08	2016-09-22	27	college	male
3	PAIDOFF	1000	30	2016-09-09	2016-10-08	28	college	female
4	PAIDOFF	1000	30	2016-09-09	2016-10-08	29	college	male

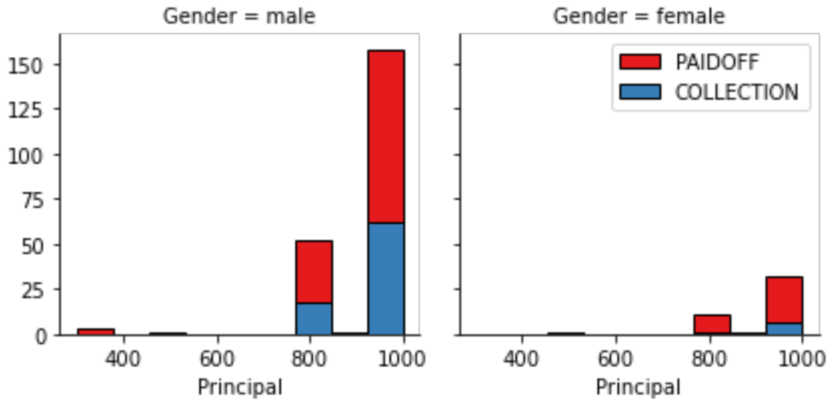
- Data visualization

In [12]:

```
bins = np.linspace(df.Principal.min(), df.Principal.max(), 10)
```

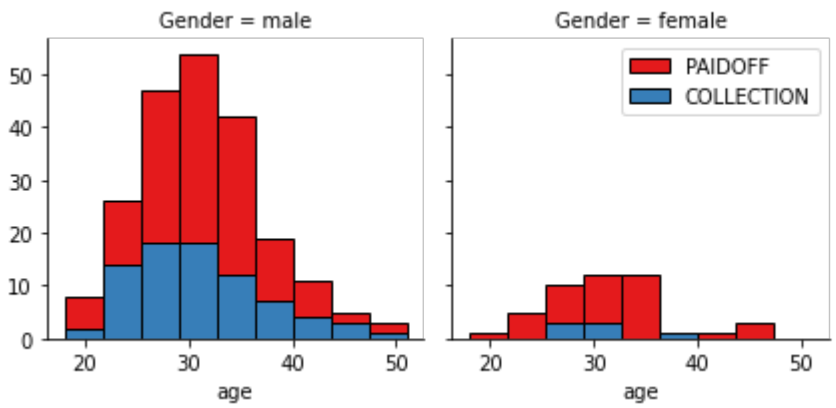
In [13]:

```
g = sns.FacetGrid(df, col="Gender", hue="loan_status", palette="Set1", col_
g.map(plt.hist, 'Principal', bins=bins, ec="k")

g.axes[-1].legend()
plt.show()
```



In [14]:

```
bins=np.linspace(df.age.min(), df.age.max(), 10)
g = sns.FacetGrid(df, col="Gender", hue="loan_status", palette="Set1", col_
g.map(plt.hist, 'age', bins=bins, ec="k")

g.axes[-1].legend()
plt.show()
```



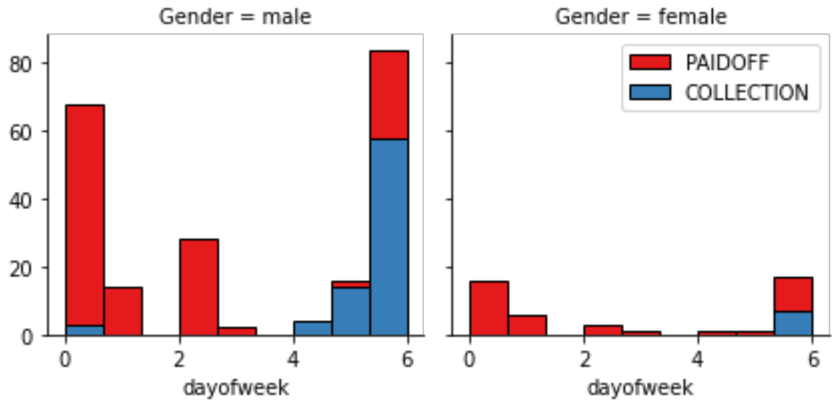
- Finding day of the week

In [15]:

```
df['dayofweek'] = df['effective_date'].dt.dayofweek
```

In [16]:

```
bins=np.linspace(df.dayofweek.min(), df.dayofweek.max(), 10)
g = sns.FacetGrid(df, col="Gender", hue="loan_status", palette="Set1", col_
g.map(plt.hist, 'dayofweek', bins=bins, ec="k")
g.axes[-1].legend()
plt.show()
```



In [17]:

```
df['weekend']= df['dayofweek'].apply(lambda x: 1 if (x>3)  else 0)
df.head()
```

Out[17]:

	loan_status	Principal	terms	effective_date	due_date	age	education	Gender	dayo
0	PAIDOFF	1000	30	2016-09-08	2016-10-07	45	High School or Below	male	
1	PAIDOFF	1000	30	2016-09-08	2016-10-07	33	Bechalor	female	
2	PAIDOFF	1000	15	2016-09-08	2016-09-22	27	college	male	
3	PAIDOFF	1000	30	2016-09-09	2016-10-08	28	college	female	
4	PAIDOFF	1000	30	2016-09-09	2016-10-08	29	college	male	

- Converting categorical features to numerical features

In [18]:

```
df['Gender'].replace(to_replace=['male','female'], value=[0,1],inplace=True
df.head()
```

Out[18]:

	loan_status	Principal	terms	effective_date	due_date	age	education	Gender	dayo

	loan_status	Principal	terms	effective_date	due_date	age	education	Gender	dayo
0	PAIDOFF	1000	30	2016-09-08	2016-10-07	45	High School or Below	0	
1	PAIDOFF	1000	30	2016-09-08	2016-10-07	33	Bechalor	1	
2	PAIDOFF	1000	15	2016-09-08	2016-09-22	27	college	0	
3	PAIDOFF	1000	30	2016-09-09	2016-10-08	28	college	1	

In [19]:

```
Feature = df[['Principal','terms','age','Gender','weekend']]
Feature = pd.concat([Feature,pd.get_dummies(df['education'])], axis=1)
Feature.drop(['Master or Above'], axis = 1,inplace=True)
Feature.head()
```

Out[19]:

	Principal	terms	age	Gender	weekend	Bechalor	High School or Below	college
0	1000	30	45	0	0	0	1	0
1	1000	30	33	1	0	1	0	0
2	1000	15	27	0	0	0	0	1
3	1000	30	28	1	1	0	0	1
4	1000	30	29	0	1	0	0	1

- Defining X and Y variables

In [20]:

```
X = Feature
```

In [21]:

```
y = df['loan_status'].values
```

- Normalizing X values

In [22]:

```
X = preprocessing.StandardScaler().fit(X).transform(X)
```

- Train and Test split

In [23]:

```
from sklearn.model_selection import train_test_split
```

In [24]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra
```

In [25]:

```
print ('Train set:', X_train.shape,  y_train.shape)
print ('Test set:', X_test.shape,  y_test.shape)
```

```
Train set: (276, 8) (276,)
Test set: (70, 8) (70,)
```

- Model 1: KNN

In [26]:

```
from sklearn.neighbors import KNeighborsClassifier
```

In [27]:

```
k=3
```

In [28]:

```
kNN_model = KNeighborsClassifier(n_neighbors=k).fit(X_train,y_train)
```

In [29]:

```
yhat = kNN_model.predict(X_test)
```

In [30]:

```
Ks=15
mean_acc=np.zeros((Ks-1))
std_acc=np.zeros((Ks-1))
ConfustionMx=[];
for n in range(1,Ks):

    #Train Model and Predict
    kNN_model = KNeighborsClassifier(n_neighbors=n).fit(X_train,y_train)
    yhat = kNN_model.predict(X_test)


    mean_acc[n-1]=np.mean(yhat==y_test);

    std_acc[n-1]=np.std(yhat==y_test)/np.sqrt(yhat.shape[0])
```

In [31]:

```
mean_acc
```

Out[31]:

```
array([0.67142857, 0.65714286, 0.71428571, 0.68571429, 0.75714286,
       0.71428571, 0.78571429, 0.75714286, 0.75714286, 0.67142857,
       0.7       , 0.72857143, 0.7       , 0.7       ])
```

K = 7 is the best

In [32]:

```
kNN_model = KNeighborsClassifier(n_neighbors=7).fit(X_train,y_train)
```

- Model 2: Decision Tree Classifier

In [33]:

```
from sklearn.tree import DecisionTreeClassifier
```

In [34]:

```
DT_model = DecisionTreeClassifier(criterion="entropy", max_depth = 4)
DT_model.fit(X_train,y_train)
```

Out[34]:

```
DecisionTreeClassifier(criterion='entropy', max_depth=4)
```

In [35]:

```
yhat = DT_model.predict(X_test)
```

- Model 3: SVC

In [36]:

```
from sklearn.svm import SVC
```

In [37]:

```
SVM_model = SVC()
SVM_model.fit(X_train, y_train)
```

Out[37]:

```
SVC()
```

In [38]:

```
yhat = SVM_model.predict(X_test)
```

- Model 4: Logistic Regression

In [39]:

```
from sklearn.linear_model import LogisticRegression
```

In [40]:

```
LR_model = LogisticRegression(C=0.01).fit(X_train,y_train)
```

In [41]:

```
yhat = LR_model.predict(X_test)
```

- Finding the best model

In [43]:

```
from sklearn.metrics import jaccard_score
from sklearn.metrics import f1_score
from sklearn.metrics import log_loss
```

- Test set

In [44]:

```
!wget -O loan_test.csv https://s3-api.us-geo.objectstorage.softlayer.net/c
```

```
--2020-11-14 21:47:39--  https://s3-api.us-geo.objectstorage.softlayer.net/
cf-courses-data/CognitiveClass/ML0101ENv3/labs/loan_test.csv
Resolvendo s3-api.us-geo.objectstorage.softlayer.net (s3-api.us-geo.objects
torage.softlayer.net)... 67.228.254.196
Conectando-se a s3-api.us-geo.objectstorage.softlayer.net (s3-api.us-geo.ob
jectstorage.softlayer.net)|67.228.254.196|:443... conectado.
A requisição HTTP foi enviada, aguardando resposta... 200 OK
Tamanho: 3642 (3,6K) [text/csv]
Salvando em: “loan_test.csv”

loan_test.csv       100%[===================>]   3,56K  --.-KB/s    em 0s

2020-11-14 21:47:40 (33,7 MB/s) - “loan_test.csv” salvo [3642/3642]
```

In [45]:

```
test_df = pd.read_csv('loan_test.csv')
```

In [46]:

```
test_df['due_date'] = pd.to_datetime(test_df['due_date'])
test_df['effective_date'] = pd.to_datetime(test_df['effective_date'])
test_df['dayofweek'] = test_df['effective_date'].dt.dayofweek
test_df['weekend'] = test_df['dayofweek'].apply(lambda x: 1 if (x>3)  else
test_df['Gender'].replace(to_replace=['male','female'], value=[0,1],inplace
test_Feature = test_df[['Principal','terms','age','Gender','weekend']]
test_Feature = pd.concat([test_Feature,pd.get_dummies(test_df['education'])
test_Feature.drop(['Master or Above'], axis = 1,inplace=True)
test_X = preprocessing.StandardScaler().fit(test_Feature).transform(test_Fe
```

In [47]:

```
test_y = test_df['loan_status'].values
```

In [48]:

```
knn_yhat = kNN_model.predict(test_X)
```

In [50]:

```
print("KNN Jaccard index: %.2f" % jaccard_score(test_y, knn_yhat, average=
print("KNN F1-score: %.2f" % f1_score(test_y, knn_yhat, average='weighted')
```

```
KNN Jaccard index: 0.38
KNN F1-score: 0.63
```

In [51]:

```
DT_yhat = DT_model.predict(test_X)
print("DT Jaccard index: %.2f" % jaccard_score(test_y, DT_yhat, average='ma
print("DT F1-score: %.2f" % f1_score(test_y, DT_yhat, average='weighted') )
```

```
DT Jaccard index: 0.53
DT F1-score: 0.74
```

In [52]:

```
SVM_yhat = SVM_model.predict(test_X)
print("SVM Jaccard index: %.2f" % jaccard_score(test_y, SVM_yhat, average=
print("SVM F1-score: %.2f" % f1_score(test_y, SVM_yhat, average='weighted')
```

```
SVM Jaccard index: 0.52
SVM F1-score: 0.76
```

In [53]:

```
LR_yhat = LR_model.predict(test_X)
LR_yhat_prob = LR_model.predict_proba(test_X)
print("LR Jaccard index: %.2f" % jaccard_score(test_y, LR_yhat, average='ma
print("LR F1-score: %.2f" % f1_score(test_y, LR_yhat, average='weighted') )
print("LR LogLoss: %.2f" % log_loss(test_y, LR_yhat_prob))
```

```
LR Jaccard index: 0.37
LR F1-score: 0.63
LR LogLoss: 0.52
```

In [ ]: