Project Title :Predicting Default Risk of Lending Club Loans

Objective:

Our main objective for this project is to implement some of the classification algorithms in order to build models to classify a good or bad loan with help of selected variables.

Dataset:

The dataset includes detailed information for every loan issued by Lending Club for year 2015.

The dataset from Lending Club contains 74 features that will be employed to train our model for prediction. Not all of the fields are intuitively useful for our learning models, such as the loan ID and the month the last payment was received, and thus we removed such fields.

We also removed fields for which greater than 80% of the values were missing.

Number of Features: 74 Number of records: 403697

Questions:

1) Which features contributes the most?

Feature Selection

1) Lasso Regression 2) Principle Component Analysis 3) Recursive Feature Elimination 4) Random Forest for Feature Ranking

Models

1) Random Forest 2) Logistic Regression 3) Support Vector Machine 4) K Nearest Neighbors(KNN)

In [49]:
```
import pandas as pd
import numpy as np
```

In [50]:
```
loans_2015 = pd.read_csv('loan2015new.csv', low_memory=False)
```

In [51]:
```
loans_2015.shape
```

Out[51]: (403697, 74)

In [52]:
```
loans_2015.columns
```

Out[52]:
```
Index(['id', 'member_id', 'loan_amnt', 'funded_amnt', 'funded_amnt_inv',
       'term', 'int_rate', 'installment', 'grade', 'sub_grade', 'emp_title',
       'emp_length', 'home_ownership', 'annual_inc', 'verification_status',
       'issue_d', 'pymnt_plan', 'url', 'desc', 'purpose', 'title', 'zip_code',
       'addr_state', 'dti', 'delinq_2yrs', 'earliest_cr_line',
       'inq_last_6mths', 'mths_since_last_delinq', 'mths_since_last_record',
       'open_acc', 'pub_rec', 'revol_bal', 'revol_util', 'total_acc',
       'initial_list_status', 'out_prncp', 'out_prncp_inv', 'total_pymnt',
       'total_pymnt_inv', 'total_rec_prncp', 'total_rec_int',
       'total_rec_late_fee', 'recoveries', 'collection_recovery_fee',
       'last_pymnt_d', 'last_pymnt_amnt', 'next_pymnt_d', 'last_credit_pull_d',
       'collections_12_mths_ex_med', 'mths_since_last_major_derog',
       'policy_code', 'application_type', 'annual_inc_joint', 'dti_joint',
       'verification_status_joint', 'acc_now_delinq', 'tot_coll_amt',
       'tot_cur_bal', 'open_acc_6m', 'open_il_6m', 'open_il_12m',
       'open_il_24m', 'mths_since_rcnt_il', 'total_bal_il', 'il_util',
       'open_rv_12m', 'open_rv_24m', 'max_bal_bc', 'all_util',
       'total_rev_hi_lim', 'inq_fi', 'total_cu_tl', 'inq_last_12m',
       'loan_status'],
      dtype='object')
```

In [53]:
```
eighty_count = len(loans_2015)*4 / 5
```

In [54]:
```
loans_2015 = loans_2015.dropna(thresh=eighty_count,axis=1)
```

In [55]:
```
loans_2015.shape
```

Out[55]:
```
(403697, 53)
```

In [56]:
```
data_dictionary = pd.read_csv('LCDataDictionary.csv')
```

In [57]:
```
data_dictionary.head()
data_dictionary = data_dictionary.rename(columns={'LoanStatNew': 'name','Descript
```

In [58]:
```
loan_df = pd.DataFrame(data=loans_2015)
```

In [59]:
```
loans_df_dtypes = pd.DataFrame(loan_df.dtypes,columns=['dtypes'])
loans_df_dtypes = loans_df_dtypes.reset_index()
loans_df_dtypes['name'] = loans_df_dtypes['index']
loans_df_dtypes = loans_df_dtypes[['name','dtypes']]
loans_df_dtypes['first value'] = loan_df.loc[0].values
preview = loans_df_dtypes.merge(data_dictionary, on='name',how='left')
```

In [60]:
```
preview
```

Out[60]:

	name	dtypes	first value	descript
0	id	int64	68587652	A unique assigned ID for loan list
1	member_id	int64	73477494	A unique assigned Id for borrower meml
2	loan_amnt	int64	25000	The listed amour the loan applied by
3	funded_amnt	int64	25000	The total amc committed to t loan at th
4	funded_amnt_inv	int64	25000	The total amc committec investors for t
5	term	object	36 months	The numbe payments on loan. Values ar
6	int_rate	float64	5.32	Interest Rate on l
7	installment	float64	752.87	The mon payment owec the borrower if t
8	grade	object	A	LC assigned l gra
9	sub_grade	object	A1	LC assigned l subgra
10	emp_title	object	Director	The job supplied by Borrower when a
11	emp_length	object	1 year	Employment ler in years. Poss values a
12	home_ownership	object	MORTGAGE	The ho ownership sta provided by bo
13	annual_inc	float64	150000	The self-repor annual inco provided by t
14	verification_status	object	Not Verified	N
15	issue_d	object	15-Dec	The month wh the loan was fun
16	pymnt_plan	object	n	Indicates payment plan been put in

	name	dtypes	first value	descript
17	url	object	https://www.lendingclub.com/browse/loanDetail....	URL for the page with lis da
18	purpose	object	credit_card	A category provi by the borrower the
19	title	object	Credit card refinancing	The loan provided by borro
20	zip_code	object	054xx	The first 3 numb of the zip c provided
21	addr_state	object	VT	The state provi by the borrowe the loa
22	dti	float64	9.54	A ratio calcula using the borrow tota
23	delinq_2yrs	int64	0	The number of days past- incidences c
24	earliest_cr_line	object	Feb-96	The month borrower's earl reported cr
25	inq_last_6mths	int64	0	The numbe inquiries in pa months (ex
26	open_acc	int64	7	The number of o credit lines in borrow
27	pub_rec	int64	0	Numbe derogatory pu reco
28	revol_bal	int64	19339	Total credit revol bala
29	revol_util	float64	42.5	Revolving utilization rate the amou
30	total_acc	int64	18	The total numbe credit lines curre i
31	initial_list_status	object	w	The initial lis status of the lo Poss
32	out_prncp	float64	24358	Remai outstand principal for t amo
33	out_prncp_inv	float64	24358	Remai outstand principal for por

	name	dtypes	first value	descript
34	total_pymnt	float64	682.67	Payments recei to date for t amount fun
35	total_pymnt_inv	float64	682.67	Payments recei to date for portio tot
36	total_rec_prncp	float64	642.03	Principal receive c
37	total_rec_int	float64	40.64	Interest receive c
38	total_rec_late_fee	float64	0	Late fees recei to c
39	recoveries	float64	0	post charge gross recov
40	collection_recovery_fee	float64	0	post charge collection
41	last_pymnt_d	object	16-Jan	Last month paym was recei
42	last_pymnt_amnt	float64	701.14	Last total paym amount recei
43	next_pymnt_d	object	16-Feb	Next schedu payment c
44	last_credit_pull_d	object	16-Jan	The most rec month LC pu credit for t
45	collections_12_mths_ex_med	int64	0	Numbe collections ir months excluc I
46	policy_code	int64	1	publicly availa policy_code=1\nr produc
47	application_type	object	INDIVIDUAL	Indicates whet the loan is individual a
48	acc_now_delinq	int64	0	The numbe accounts on wh the borrowe
49	tot_coll_amt	int64	0	Total collec amounts ever ov
50	tot_cur_bal	int64	430856	Total curr balance o accou
51	total_rev_hi_lim	int64	45500	N
52	loan_status	object	Current	Current status of l

In [61]:
```
preview[:19]
```

Out[61]:

	name	dtypes	first value	description
0	id	int64	68587652	A unique LC assigned ID for the loan listing.
1	member_id	int64	73477494	A unique LC assigned Id for the borrower member.
2	loan_amnt	int64	25000	The listed amount of the loan applied for by t...
3	funded_amnt	int64	25000	The total amount committed to that loan at tha...
4	funded_amnt_inv	int64	25000	The total amount committed by investors for th...
5	term	object	36 months	The number of payments on the loan. Values are...
6	int_rate	float64	5.32	Interest Rate on the loan
7	installment	float64	752.87	The monthly payment owed by the borrower if th...
8	grade	object	A	LC assigned loan grade
9	sub_grade	object	A1	LC assigned loan subgrade
10	emp_title	object	Director	The job title supplied by the Borrower when ap...
11	emp_length	object	1 year	Employment length in years. Possible values ar...
12	home_ownership	object	MORTGAGE	The home ownership status provided by the borr...
13	annual_inc	float64	150000	The self-reported annual income provided by th...
14	verification_status	object	Not Verified	NaN
15	issue_d	object	15-Dec	The month which the loan was funded
16	pymnt_plan	object	n	Indicates if a payment plan has been put in pl...
17	url	object	https://www.lendingclub.com/browse/loanDetail....	URL for the LC page with listing data.
18	purpose	object	credit_card	A category provided by the borrower for the lo...

In [62]:
```
drop_list = ['id','member_id','funded_amnt','funded_amnt_inv',
             'int_rate','sub_grade','emp_title','issue_d','url']
loan_df = loan_df.drop(drop_list,axis=1)
```

In [63]:
```
loan_df.shape
```

Out[63]: (403697, 44)

In [64]:
```
preview[19:38]
```

Out[64]:

	name	dtypes	first value	description
19	title	object	Credit card refinancing	The loan title provided by the borrower
20	zip_code	object	054xx	The first 3 numbers of the zip code provided b...
21	addr_state	object	VT	The state provided by the borrower in the loan...
22	dti	float64	9.54	A ratio calculated using the borrower’s total ...
23	delinq_2yrs	int64	0	The number of 30+ days past-due incidences of ...
24	earliest_cr_line	object	Feb-96	The month the borrower's earliest reported cre...
25	inq_last_6mths	int64	0	The number of inquiries in past 6 months (excl...
26	open_acc	int64	7	The number of open credit lines in the borrowe...
27	pub_rec	int64	0	Number of derogatory public records
28	revol_bal	int64	19339	Total credit revolving balance
29	revol_util	float64	42.5	Revolving line utilization rate, or the amount...
30	total_acc	int64	18	The total number of credit lines currently in ...
31	initial_list_status	object	w	The initial listing status of the loan. Possib...
32	out_prncp	float64	24358	Remaining outstanding principal for total amou...
33	out_prncp_inv	float64	24358	Remaining outstanding principal for portion of...
34	total_pymnt	float64	682.67	Payments received to date for total amount funded
35	total_pymnt_inv	float64	682.67	Payments received to date for portion of total...
36	total_rec_prncp	float64	642.03	Principal received to date
37	total_rec_int	float64	40.64	Interest received to date

In [65]:
```
drop_cols = ['zip_code','out_prncp','out_prncp_inv',
             'total_pymnt','total_pymnt_inv']
loan_df = loan_df.drop(drop_cols, axis=1)
```

In [66]:
```
loan_df.shape
```

Out[66]: (403697, 39)

In [67]:
```
preview[38:]
```

Out[67]:

	name	dtypes	first value	description
38	total_rec_late_fee	float64	0	Late fees received to date
39	recoveries	float64	0	post charge off gross recovery
40	collection_recovery_fee	float64	0	post charge off collection fee
41	last_pymnt_d	object	16-Jan	Last month payment was received
42	last_pymnt_amnt	float64	701.14	Last total payment amount received
43	next_pymnt_d	object	16-Feb	Next scheduled payment date
44	last_credit_pull_d	object	16-Jan	The most recent month LC pulled credit for thi...
45	collections_12_mths_ex_med	int64	0	Number of collections in 12 months excluding m...
46	policy_code	int64	1	publicly available policy_code=1\nnew products...
47	application_type	object	INDIVIDUAL	Indicates whether the loan is an individual ap...
48	acc_now_delinq	int64	0	The number of accounts on which the borrower i...
49	tot_coll_amt	int64	0	Total collection amounts ever owed
50	tot_cur_bal	int64	430856	Total current balance of all accounts
51	total_rev_hi_lim	int64	45500	NaN
52	loan_status	object	Current	Current status of the loan

In [68]:
```
drop_cols = ['total_rec_prncp','total_rec_int', 'total_rec_late_fee',
             'recoveries', 'collection_recovery_fee', 'last_pymnt_d']

loan_df = loan_df.drop(drop_cols, axis=1)
```

In [69]:
```
loan_df.shape
```

Out[69]:
```
(403697, 33)
```

In [70]:
```
loan_df["loan_status"].value_counts()
```

Out[70]:
```
Current        377553
Fully Paid      22984
Charged Off      2773
Default           387
Name: loan_status, dtype: int64
```

In [71]:
```
loan_df = loan_df[(loan_df["loan_status"] == "Fully Paid") |
                            (loan_df["loan_status"] == "Charged Off")]

mapping_dictionary = {"loan_status":{ "Fully Paid": 1, "Charged Off": 0}}
loan_df = loan_df.replace(mapping_dictionary)
```

In [73]:
```
loan_df = loan_df.loc[:,loan_df.apply(pd.Series.nunique) != 1]
```

In [74]:
```
for col in loan_df.columns:
    if (len(loan_df[col].unique()) < 4):
        print(loan_df[col].value_counts())
        print()
```

```
 36 months    18102
 60 months     7655
Name: term, dtype: int64

MORTGAGE    12824
RENT        10026
OWN          2907
Name: home_ownership, dtype: int64

Source Verified    10678
Verified            8132
Not Verified        6947
Name: verification_status, dtype: int64

w    13615
f    12142
Name: initial_list_status, dtype: int64

Series([], Name: next_pymnt_d, dtype: int64)

INDIVIDUAL    25756
JOINT             1
Name: application_type, dtype: int64

0    25612
1      135
2       10
Name: acc_now_delinq, dtype: int64

1    22984
0     2773
Name: loan_status, dtype: int64
```

In [75]:
```
print(loan_df.shape[1])
loan_df = loan_df.drop('application_type', axis=1)
print("We've been able to reduced the features to => {}".format(loan_df.shape[1])
```

```
31
We've been able to reduced the features to => 30
```

In [76]:
```
loan_df = loan_df[(loan_df["initial_list_status"] == "w") |
                            (loan_df["initial_list_status"] == "f")]

mapping_dictionary = {"initial_list_status":{ "w": 1, "f": 0}}
loan_df = loan_df.replace(mapping_dictionary)
```

In [77]:
```
loan_df.to_csv("filtered_loans_2015.csv",index=False)
```

Handle missing values and categorical features before feeding the data into a machine learning algorithm

In [78]:
```
filtered_loans = pd.read_csv('filtered_loans_2015.csv')
print(filtered_loans.shape)
filtered_loans.head()
```

```
(25757, 30)
```

Out[78]:

	loan_amnt	term	installment	grade	emp_length	home_ownership	annual_inc	verification_sta
0	19800	36 months	666.00	C	7 years	MORTGAGE	78924.0	Not Ver
1	35000	36 months	1177.27	C	10+ years	RENT	95000.0	Source Ver
2	20000	36 months	672.73	C	6 years	MORTGAGE	56000.0	Not Ver
3	28000	60 months	635.37	C	5 years	MORTGAGE	96000.0	Not Ver
4	23100	60 months	605.35	E	10+ years	MORTGAGE	55000.0	Source Ver

5 rows × 30 columns

In [79]:
```
null_counts = filtered_loans.isnull().sum()
print("Number of null values in each column:\n{}".format(null_counts))
```

```
Number of null values in each column:
loan_amnt                         0
term                              0
installment                       0
grade                             0
emp_length                        0
home_ownership                    0
annual_inc                        0
verification_status               0
purpose                           0
title                             0
addr_state                        0
dti                               0
delinq_2yrs                       0
earliest_cr_line                  0
inq_last_6mths                    0
open_acc                          0
pub_rec                           0
revol_bal                         0
revol_util                       13
total_acc                         0
initial_list_status               0
last_pymnt_amnt                   0
next_pymnt_d                  25757
last_credit_pull_d                0
collections_12_mths_ex_med        0
acc_now_delinq                    0
tot_coll_amt                      0
tot_cur_bal                       0
total_rev_hi_lim                  0
loan_status                       0
dtype: int64
```

Remove entire column next_pymnt_d column as data is missing for entire filtered loans dataset. Drop the missing rows for revol_util

In [80]:
```
filtered_loans = filtered_loans.drop("next_pymnt_d",axis=1)
filtered_loans = filtered_loans.dropna()
```

In [81]:
```
filtered_loans.shape
```

Out[81]: (25744, 29)

Convert categorical values to numerical

In [82]:
```
print("Data types and their frequency\n{}".format(filtered_loans.dtypes.value_cou
```

```
Data types and their frequency
int64      14
object     10
float64     5
dtype: int64
```

In [83]:
```
object_columns_df = filtered_loans.select_dtypes(include=['object'])
print(object_columns_df.iloc[0])
```

```
term                            36 months
grade                                   C
emp_length                        7 years
home_ownership                   MORTGAGE
verification_status          Not Verified
purpose                debt_consolidation
title                  Debt consolidation
addr_state                             MD
earliest_cr_line                    5-Oct
last_credit_pull_d                 16-Jan
Name: 0, dtype: object
```

In [84]:
```
cols = ['home_ownership', 'grade','verification_status', 'emp_length', 'term', 'a
for name in cols:
    print(name,':')
    print(object_columns_df[name].value_counts(),'\n')
```

```
home_ownership :
MORTGAGE    12816
RENT        10022
OWN          2906
Name: home_ownership, dtype: int64

grade :
C    7576
B    5945
D    4663
A    3314
E    2906
F    1048
G     292
Name: grade, dtype: int64

verification_status :
Source Verified    10671
Verified            8128
Not Verified        6945
Name: verification_status, dtype: int64

emp_length :
10+ years    8850
2 years      2321
< 1 year     2095
3 years      1952
1 year       1694
4 years      1504
5 years      1460
8 years      1311
n/a          1224
7 years      1199
6 years      1092
9 years      1042
Name: emp_length, dtype: int64

term :
 36 months    18092
 60 months     7652
Name: term, dtype: int64

addr_state :
CA    4153
TX    2068
NY    1862
FL    1772
IL     890
NJ     842
GA     824
PA     808
```

```
OH     800
VA     795
NC     763
MI     715
AZ     666
CO     659
WA     623
MD     620
MA     526
MN     512
NV     430
IN     423
MO     367
OR     362
TN     362
WI     347
AL     327
LA     312
CT     289
SC     272
UT     247
OK     223
KY     223
KS     200
HI     172
AR     164
NM     151
MS     139
RI     119
WV     119
NH     105
DE      87
MT      73
DC      66
WY      60
AK      57
SD      56
VT      46
NE      31
ME       9
ND       8
Name: addr_state, dtype: int64
```

In [85]:
```
for name in ['purpose','title']:
    print("Unique Values in column: {}\n".format(name))
    print(filtered_loans[name].value_counts(),'\n')
```

```
Unique Values in column: purpose

debt_consolidation    16080
credit_card            4526
home_improvement       1725
other                  1442
major_purchase          545
medical                 275
small_business          260
car                     255
moving                  231
house                   203
vacation                185
renewable_energy         16
wedding                   1
Name: purpose, dtype: int64

Unique Values in column: title

Debt consolidation         16072
Credit card refinancing     4535
Home improvement            1726
Other                       1442
Major purchase               543
Medical expenses             275
Business                     259
Car financing                255
Moving and relocation        231
Home buying                  204
Vacation                     185
Green loan                    16
Credit Card/Auto Repair        1
Name: title, dtype: int64
```

The purpose and title columns do contain overlapping information, but the purpose column contains fewer discrete values and is cleaner, so we'll keep it and drop title. The addr_state column, however,contains too many unique values, so it's better to drop this

In [86]:
```
drop_cols = ['last_credit_pull_d','addr_state','title','earliest_cr_line','purpos
filtered_loans = filtered_loans.drop(drop_cols,axis=1)
```

Convert Ordinal values to numerical values

In [87]:
```
filtered_loans['grade'] = filtered_loans['grade'].map({'A':7,'B':6,'C':5,'D':4,'E
filtered_loans["home_ownership"] = filtered_loans["home_ownership"].map({"MORTGAG
filtered_loans["emp_length"] = filtered_loans["emp_length"].replace({'years':'','
filtered_loans["emp_length"] = filtered_loans["emp_length"].apply(lambda x:int(x)
filtered_loans['verification_status'] = filtered_loans['verification_status'].map
filtered_loans["term"] = filtered_loans["term"].map({" 36 months":0," 60 months":
print("Current shape of dataset :",filtered_loans.shape)
filtered_loans.head()
```

```
Current shape of dataset : (25744, 24)
```

Out[87]:

	loan_amnt	term	installment	grade	emp_length	home_ownership	annual_inc	verification_statu
0	19800	0	666.00	5	7	3	78924.0	
1	35000	0	1177.27	5	10	2	95000.0	
2	20000	0	672.73	5	6	3	56000.0	
3	28000	1	635.37	5	5	3	96000.0	
4	23100	1	605.35	3	10	3	55000.0	

5 rows × 24 columns

Convert Nominal values to numeric values

In [88]:
```
filtered_loans.to_csv("cleaned_loans_2015.csv",index=False)
```

In [89]:
```
filtered_loans.shape
```

Out[89]: (25744, 24)

In [90]:
```
X = filtered_loans.iloc[:,0:22]
y = filtered_loans.iloc[:,-1]
```

In [91]:
```
X.head()
```

Out[91]:

	loan_amnt	term	installment	grade	emp_length	home_ownership	annual_inc	verification_statu
0	19800	0	666.00	5	7	3	78924.0	
1	35000	0	1177.27	5	10	2	95000.0	
2	20000	0	672.73	5	6	3	56000.0	
3	28000	1	635.37	5	5	3	96000.0	
4	23100	1	605.35	3	10	3	55000.0	

5 rows × 22 columns

In [92]:
```
y.head()
```

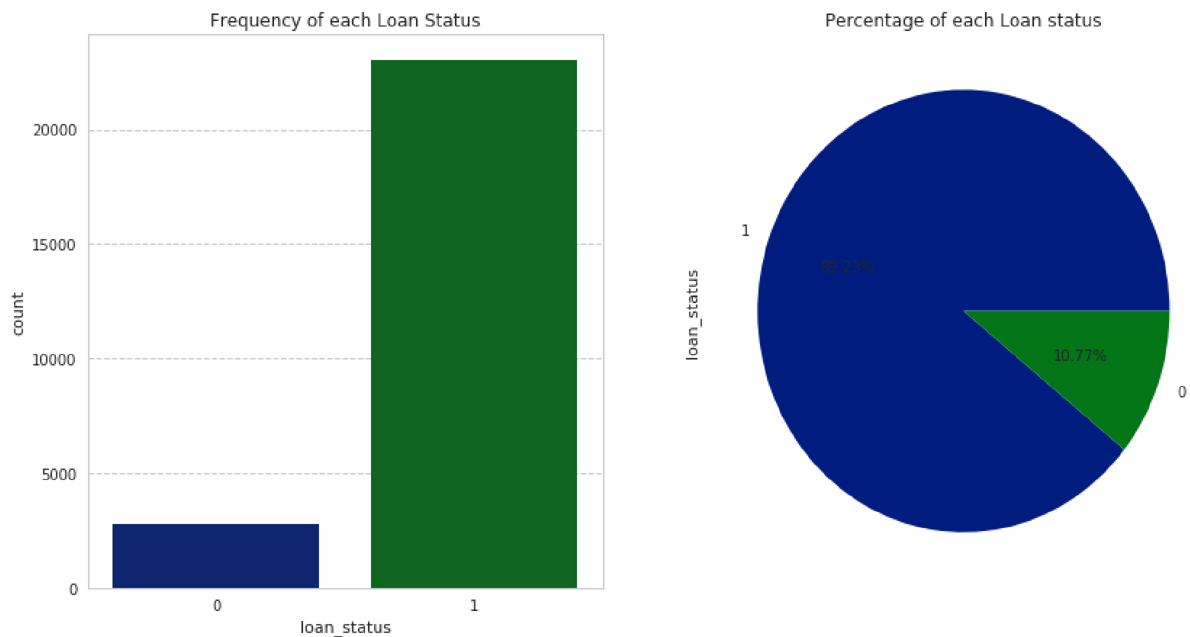
Out[92]:
```
0    1
1    1
2    1
3    1
4    1
Name: loan_status, dtype: int64
```

In [93]:
```
from sklearn import preprocessing,metrics
from IPython.core.display import HTML

filtered_loans.fillna(filtered_loans.mean(),inplace = True)
HTML(filtered_loans.tail().to_html())
print("Current shape of dataset :",filtered_loans.shape)
```

```
Current shape of dataset : (25744, 24)
```

In [166]:
```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
plt.rcParams['figure.figsize'] = (12,8)

fig, axs = plt.subplots(1,2,figsize=(14,7))
sns.countplot(x='loan_status',data=loan_df,ax=axs[0])
axs[0].set_title("Frequency of each Loan Status")
loan_df.loan_status.value_counts().plot(x=None,y=None, kind='pie', ax=axs[1],auto
axs[1].set_title("Percentage of each Loan status")
plt.show()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\font_manager.py:1316: Use
rWarning: findfont: Font family ['Ricty'] not found. Falling back to DejaVu San
s
  (prop.get_family(), self.defaultFamily[fontext]))
```



In [94]:
```
scl = preprocessing.StandardScaler() #instance of preprocessing
fields = filtered_loans.columns.values[:-1]
data_clean = pd.DataFrame(scl.fit_transform(filtered_loans[fields]), columns = fi
data_clean['loan_status'] = filtered_loans['loan_status']
data_clean['loan_status'].value_counts()
```

Out[94]:
```
1.0    22964
0.0     2767
Name: loan_status, dtype: int64
```

In [95]:
```
from sklearn.utils import resample
loanstatus_0 = data_clean[data_clean["loan_status"]==0]
loanstatus_1 = data_clean[data_clean["loan_status"]==1]
loanstatus_0_upsampled = resample(loanstatus_0,
                                replace=True,     # sample with replacement
                                n_samples=20980,    # to match majority class
                                random_state=123) # reproducible results
data_clean = pd.concat([loanstatus_1, loanstatus_0_upsampled ])
data_clean = data_clean.sample(frac=1).reset_index(drop=True)
print("Current shape of dataset :",data_clean.shape)

loanstatus_0_upsampled.shape
```

```
Current shape of dataset : (43944, 24)
```

Out[95]:
```
(20980, 24)
```

In [98]:
```
data_clean.corr()
```

Out[98]:

	loan_amnt	term	installment	grade	emp_length	home_owne
loan_amnt	1.000000	0.455861	0.952071	-0.225756	0.099306	0.12
term	0.455861	1.000000	0.217305	-0.476183	0.064707	0.06
installment	0.952071	0.217305	1.000000	-0.217058	0.084692	0.10
grade	-0.225756	-0.476183	-0.217058	1.000000	0.019103	0.06
emp_length	0.099306	0.064707	0.084692	0.019103	1.000000	0.13
home_ownership	0.129773	0.068420	0.109254	0.063040	0.131661	1.00
annual_inc	0.317929	0.073557	0.306493	0.067256	0.087671	0.11
verification_status	-0.216073	-0.176614	-0.196829	0.144564	-0.015045	0.04
dti	-0.017879	0.063224	-0.015202	-0.183153	0.013582	-0.04
delinq_2yrs	-0.011711	-0.013231	-0.005467	-0.038706	0.017797	0.02
inq_last_6mths	-0.020390	-0.001253	0.011767	-0.215075	-0.001954	-0.00
open_acc	0.175538	0.080293	0.163494	-0.002158	0.035108	0.07
pub_rec	-0.123417	-0.071160	-0.102815	-0.047983	-0.004923	-0.00
revol_bal	0.316639	0.098604	0.304522	0.008651	0.072913	0.10
revol_util	0.117621	0.095156	0.120929	-0.194006	0.052414	0.05
total_acc	0.184690	0.090346	0.163928	0.042440	0.110806	0.12
initial_list_status	0.082138	0.166243	0.011267	0.165295	0.035411	0.05
last_pymnt_amnt	0.725848	0.325575	0.679022	-0.104151	0.101745	0.13
collections_12_mths_ex_med	0.000159	0.003060	0.003543	-0.026857	-0.003952	-0.00
acc_now_delinq	0.024120	0.016880	0.023888	-0.037193	0.005935	-0.01
tot_coll_amt	-0.026682	-0.001487	-0.025481	-0.006535	0.018920	0.00
tot_cur_bal	0.332941	0.126096	0.305926	0.052522	0.108377	0.39
total_rev_hi_lim	0.352104	0.088110	0.327118	0.138139	0.070930	0.10
loan_status	0.010869	-0.010601	0.012866	0.021460	0.006428	-0.00

24 rows × 24 columns

In [99]:
```
import numpy as np
from sklearn.svm import SVC
from sklearn.model_selection import learning_curve
from sklearn.model_selection import ShuffleSplit
from sklearn import linear_model,svm
from sklearn.metrics import average_precision_score
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split

def plot_learning_curve(estimator, title, X, y, ylim=None, cv=None,
                        n_jobs=1, train_sizes=np.linspace(.1, 1.0, 5)):
    plt.figure()
    plt.title(title)
    if ylim is not None:
        plt.ylim(*ylim)
    plt.xlabel("Training examples")
    plt.ylabel("Score")
    train_sizes, train_scores, test_scores = learning_curve(estimator, X, y, cv=c
    train_scores_mean = np.mean(train_scores, axis=1)
    train_scores_std = np.std(train_scores, axis=1)
    test_scores_mean = np.mean(test_scores, axis=1)
    test_scores_std = np.std(test_scores, axis=1)
    plt.grid()

    plt.fill_between(train_sizes, train_scores_mean - train_scores_std,
                     train_scores_mean + train_scores_std, alpha=0.1,
                     color="r")
    plt.fill_between(train_sizes, test_scores_mean - test_scores_std,
                     test_scores_mean + test_scores_std, alpha=0.1, color="g")
    plt.plot(train_sizes, train_scores_mean, 'o-', color="r",
             label="Training score")
    plt.plot(train_sizes, test_scores_mean, 'o-', color="g",
             label="Cross-validation score")
    plt.legend(loc="best")
    return plt

X, y = data_clean.iloc[:,:-1].values, data_clean.iloc[:,-1].values
title = "Learning Curves (Logistic Regression)"
# Cross validation with 100 iterations to get smoother mean test and train
# score curves, each time with 20% data randomly selected as a validation set.
cv = ShuffleSplit(n_splits=100, test_size=0.2, random_state=0)
estimator = linear_model.LogisticRegression()
plot_learning_curve(estimator, title, X, y, ylim=(0.15, 0.90), cv=cv, n_jobs=4)
plt.show()
```

In [100]:

```
import seaborn as sns
sns.set('talk', 'whitegrid', 'dark', font_scale=1, font='Ricty',rc={"lines.linewi
def plotAUC(truth, pred, lab):
    fpr, tpr, _ = metrics.roc_curve(truth,pred)
    roc_auc = metrics.auc(fpr, tpr)
    lw = 2
    c = (np.random.rand(), np.random.rand(), np.random.rand())
    plt.plot(fpr, tpr, color= c,lw=lw, label= lab +'(AUC = %0.2f)' % roc_auc)
    plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.0])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('ROC curve') #Receiver Operating Characteristic
    plt.legend(loc="lower right")
```

In [101]:
```
import itertools
from sklearn.metrics import confusion_matrix
def plot_confusion_matrix(model, normalize=False): # This function prints and plo
    cm = confusion_matrix(y_test, model, labels=[0, 1])
    classes=["Will Pay", "Will Default"]
    cmap = plt.cm.Blues
    title = "Confusion Matrix"
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        cm = np.around(cm, decimals=3)
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")
    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
```

In [102]:
```
X_train, X_test, y_train, y_test = train_test_split(data_clean.iloc[:,:-1], data_
bs_train, bs_test = train_test_split(data_clean, test_size = 0.2, random_state=42
```

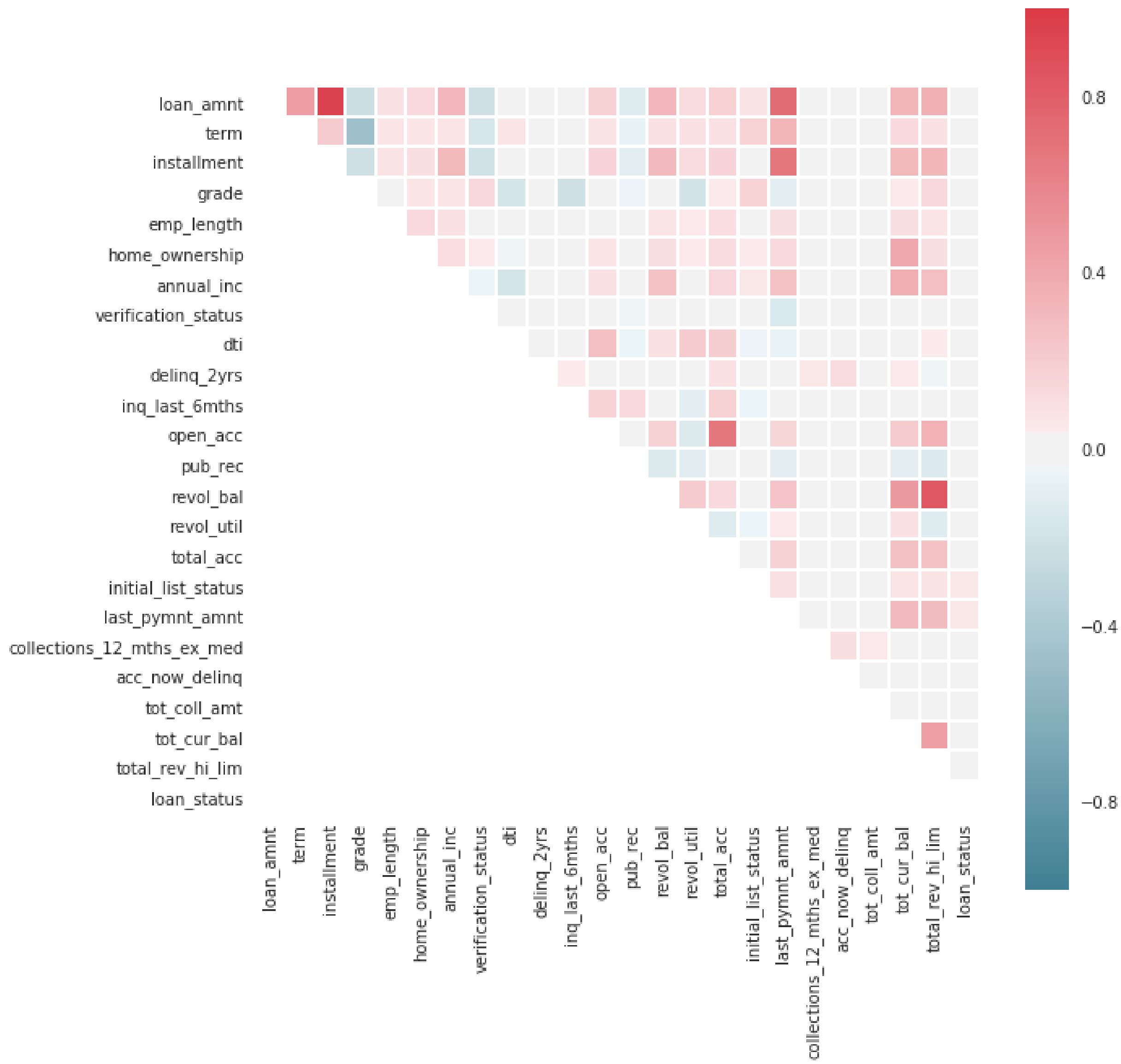
In [103]:
```
#PCA (Principal Component Analysis)
from sklearn.decomposition import PCA
pca = PCA(n_components=10, whiten=True)
X_train_pca = pca.fit_transform(X_train)
X_test_pca = pca.transform(X_test)
explained_variance = pca.explained_variance_ratio_
print('Expected Variance is '+ str(explained_variance))
```

```
Expected Variance is [ 0.17256429  0.08791379  0.07494317  0.06342185  0.057395
6   0.05214995
  0.05023233  0.04913359  0.04449675  0.04376371]
```

In [104]:
```
dataViz = data_clean
sns.set_context(context='notebook')
fig, ax = plt.subplots(figsize=(10,10))
corr = dataViz.corr()

# Generate a mask for the upper triangle
mask = np.zeros_like(corr, dtype=np.bool)
mask[np.tril_indices_from(mask)] = True

# Generate a custom diverging colormap
cmap = sns.diverging_palette(220, 10, as_cmap=True)

sns.heatmap(corr, cmap=cmap,linewidths=1, vmin=-1, vmax=1, square=True, cbar=True
```

```
C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\font_manager.py:1316: Use
rWarning: findfont: Font family ['Ricty'] not found. Falling back to DejaVu San
s
  (prop.get_family(), self.defaultFamily[fontext]))
```

Out[104]: <matplotlib.axes._subplots.AxesSubplot at 0x2a91df4c518>

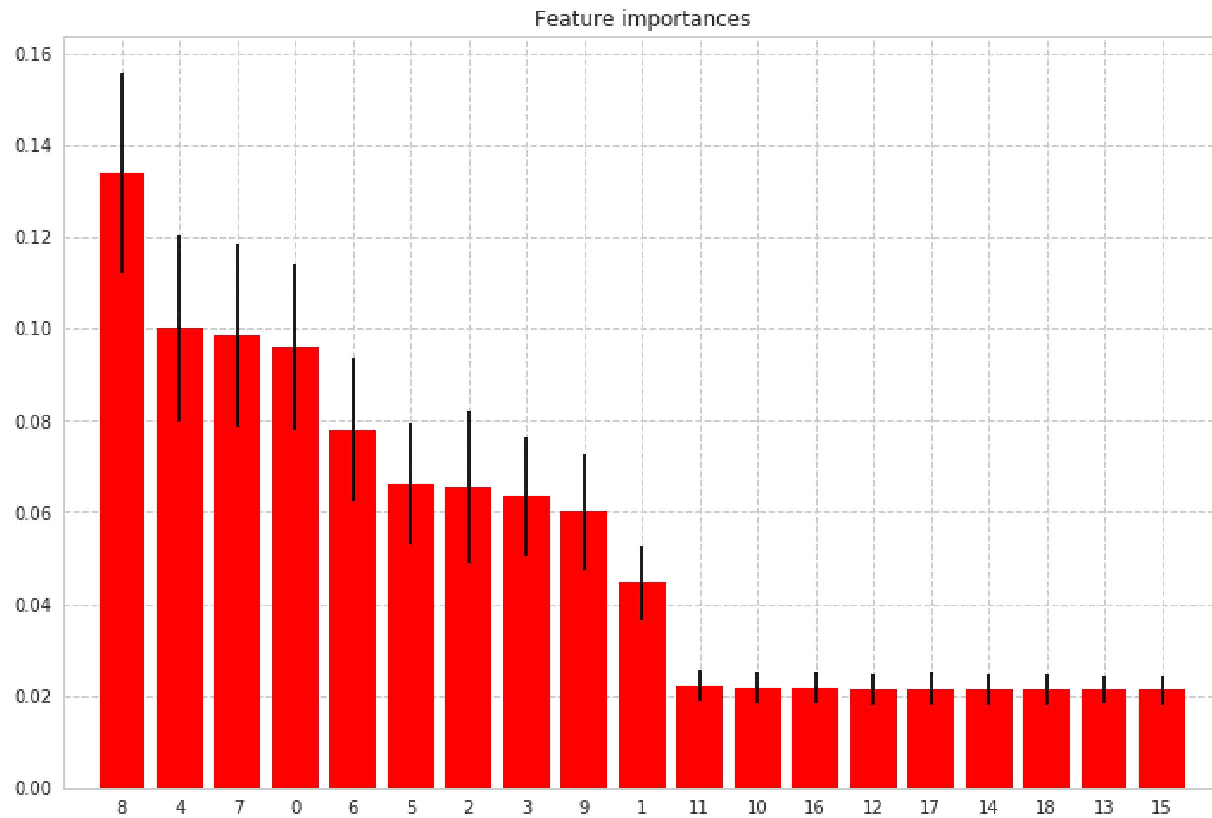


In [105]:
```
from sklearn.datasets import make_classification
from sklearn.ensemble import ExtraTreesClassifier

# Build a classification task using 3 informative features
X, y = make_classification(n_samples=6767,
                           n_features=19,
                           n_informative=10,
                           n_redundant=0,
                           n_repeated=0,
                           n_classes=2,
                           random_state=0,
                           shuffle=False)

# Build a forest and compute the feature importances
forest = ExtraTreesClassifier(n_estimators=250,
                              random_state=0)

forest.fit(X, y)
importances = forest.feature_importances_
std = np.std([tree.feature_importances_ for tree in forest.estimators_],
             axis=0)
indices = np.argsort(importances)[::-1]


# Print the feature ranking
print("Feature ranking:")

for f in range(X.shape[1]):
    print("%d. feature %d (%f)" % (f + 1, indices[f], importances[indices[f]]))

# Plot the feature importances of the forest
plt.figure()
plt.title("Feature importances")
plt.bar(range(X.shape[1]), importances[indices],
       color="r", yerr=std[indices], align="center")
plt.xticks(range(X.shape[1]), indices)
plt.xlim([-1, X.shape[1]])
plt.show()
```

```
Feature ranking:
1. feature 8 (0.134019)
2. feature 4 (0.099906)
3. feature 7 (0.098578)
4. feature 0 (0.095823)
5. feature 6 (0.077986)
6. feature 5 (0.066192)
7. feature 2 (0.065415)
8. feature 3 (0.063437)
9. feature 9 (0.059985)
10. feature 1 (0.044657)
11. feature 11 (0.022180)
12. feature 10 (0.021810)
13. feature 16 (0.021685)
14. feature 12 (0.021542)
15. feature 17 (0.021533)
16. feature 14 (0.021354)
```

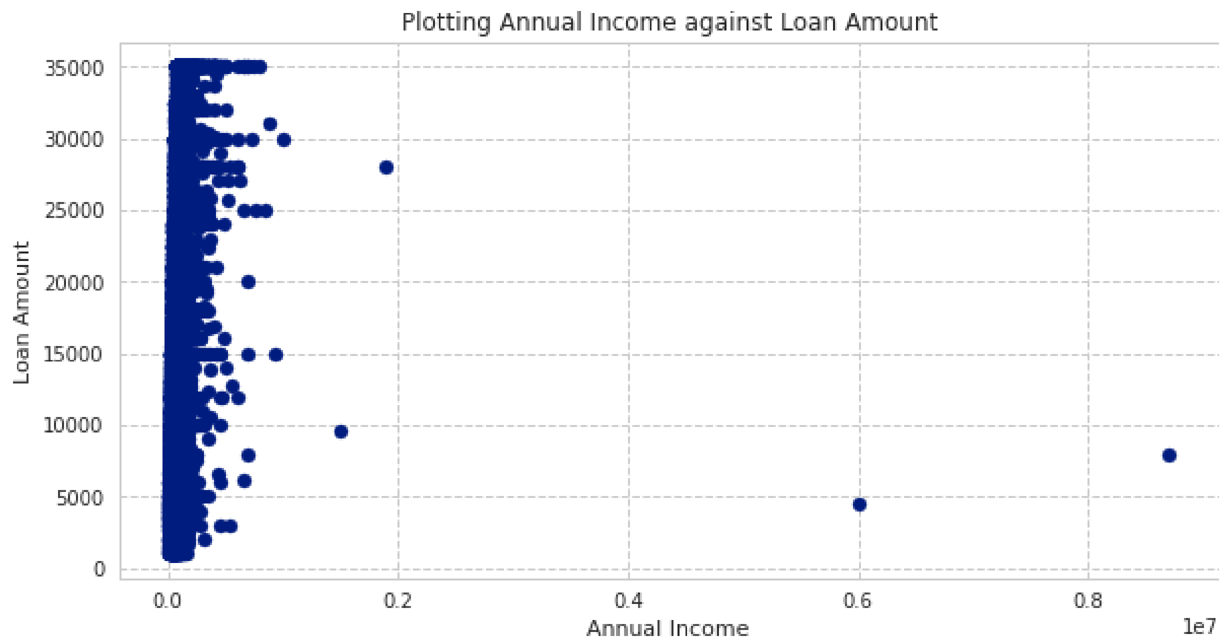
```
17. feature 18 (0.021346)
18. feature 13 (0.021307)
19. feature 15 (0.021246)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\font_manager.py:1316: Use
rWarning: findfont: Font family ['Ricty'] not found. Falling back to DejaVu San
s
  (prop.get_family(), self.defaultFamily[fontext]))
```

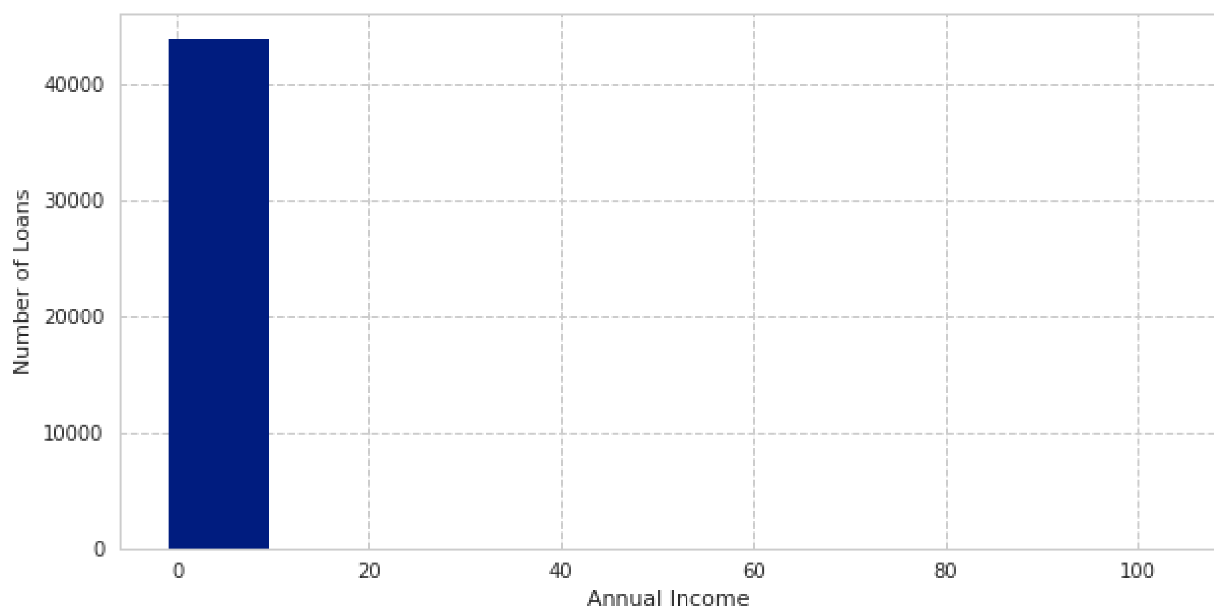


In [132]:
```
plt.figure(figsize=(10,5))
plt.scatter(filtered_loans['annual_inc'], filtered_loans['loan_amnt'])
plt.title("Plotting Annual Income against Loan Amount")
plt.ylabel('Loan Amount')
plt.xlabel('Annual Income')
plt.show()

data_clean.annual_inc.hist(figsize=(10,5))
plt.ylabel('Number of Loans')
plt.xlabel('Annual Income')
```

```
C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\font_manager.py:1316: Use
rWarning: findfont: Font family ['Ricty'] not found. Falling back to DejaVu San
s
  (prop.get_family(), self.defaultFamily[fontext]))
```



Out[132]: Text(0.5,0,'Annual Income')



In [133]:
```
data_clean = data_clean[data_clean['annual_inc']<200000]
```

In [135]:
```
X_Variables = ['emp_length', 'grade']
X = data_clean[X_Variables]
X = X.values
y = data_clean['loan_status'].values
clf = linear_model.LogisticRegression()
model = clf.fit(X,y)
model.score(X, y)
pd.DataFrame(list(zip(X_Variables,model.coef_.T)))
```

Out[135]:

	0	1
0	emp_length	[0.0120505219261]
1	grade	[0.0426461226429]

Lets take a look at the co-efficient of grade a loan. For every additional increase in the grade "G" to "F" or in our case "1" to "2" the chance of the loan being paid off increases by 0.04 and it shows increase in emp_length by factor of 0.012

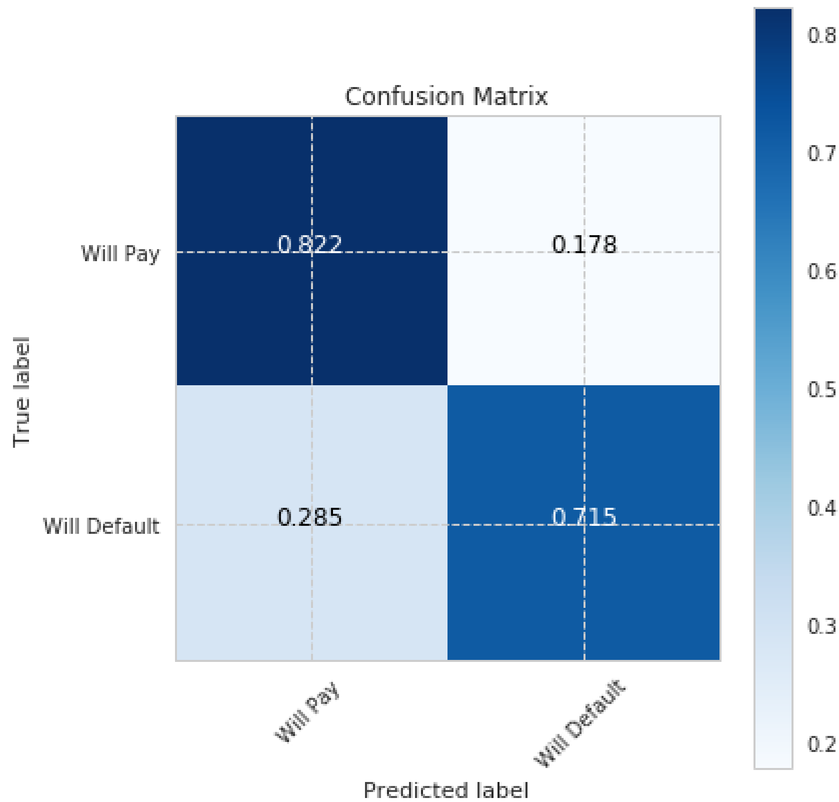
In [150]:
```
from sklearn.model_selection import GridSearchCV
def cross_validation_best_parameters(model, param_grid):
    grid = GridSearchCV(model, param_grid,cv=10, scoring='accuracy')
    X=data_clean.iloc[:,:-1].values
    y=data_clean.iloc[:,-1].values
    grid.fit(X,y)
    mean_scores = [result.mean_validation_score for result in grid.grid_scores_]
    return mean_scores,grid.best_score_,grid.best_estimator_
logreg = linear_model.LogisticRegression(random_state=0)
c=[0.001, 0.01, 0.1, 1, 10, 100, 1000]
param_grid = dict(C=c)
mean_scores,Best_Accuracy, Best_classifier = cross_validation_best_parameters(log
print(Best_classifier)
```

```
LogisticRegression(C=0.01, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=0, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\model_selection\_search.py:7
61: DeprecationWarning: The grid_scores_ attribute was deprecated in version 0.
18 in favor of the more elaborate cv_results_ attribute. The grid_scores_ attri
bute will not be available from 0.20
  DeprecationWarning)
```

In [149]:
```
clf_LR = linear_model.LogisticRegression(C=0.01)
clf_LR.fit(X_train,y_train)
LR_Predict = clf_LR.predict_proba(X_test)[:,1]
LR_Predict_bin = clf_LR.predict(X_test)
LR_Accuracy = accuracy_score(y_test,LR_Predict.round())
print("Logistic regression accuracy is ",LR_Accuracy)
plt.figure(figsize=(6,6))
plot_confusion_matrix(LR_Predict_bin, normalize=True)
plt.show()
```

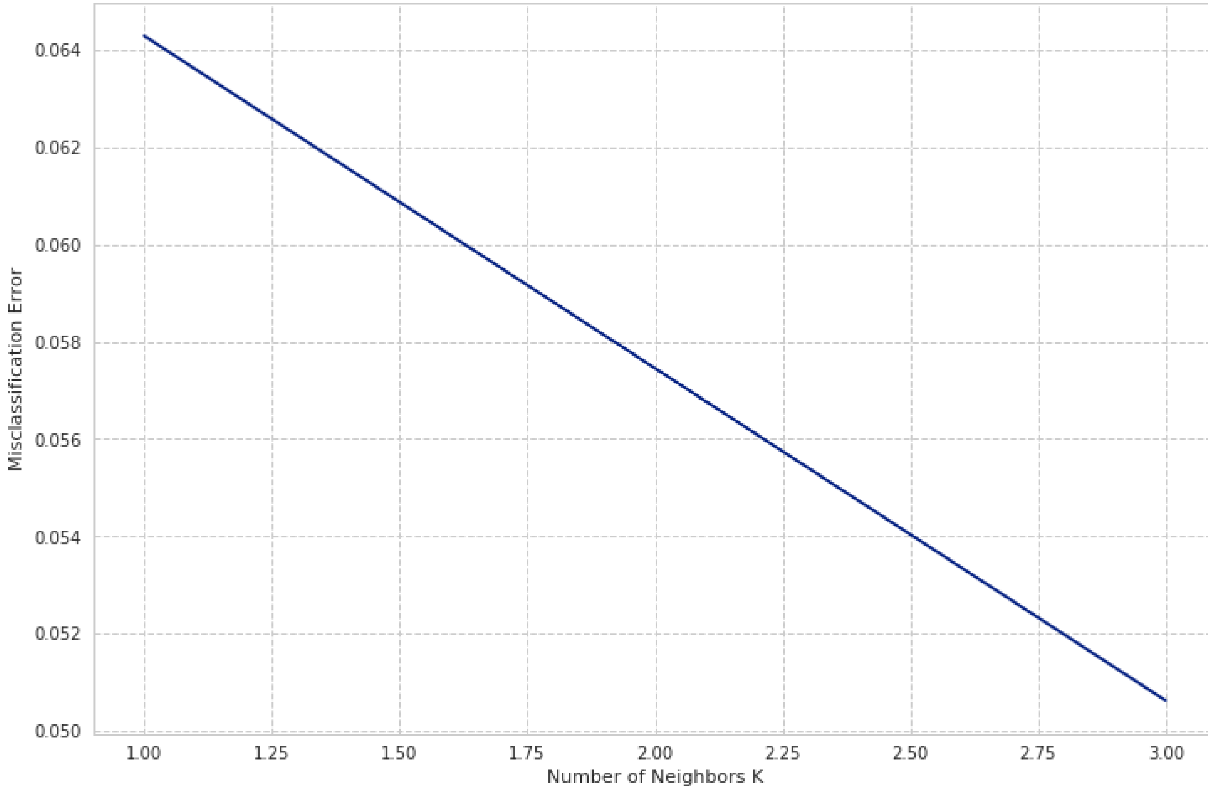
```
Logistic regression accuracy is  0.768094534712
```

```
C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\font_manager.py:1316: Use
rWarning: findfont: Font family ['Ricty'] not found. Falling back to DejaVu San
s
  (prop.get_family(), self.defaultFamily[fontext]))
```



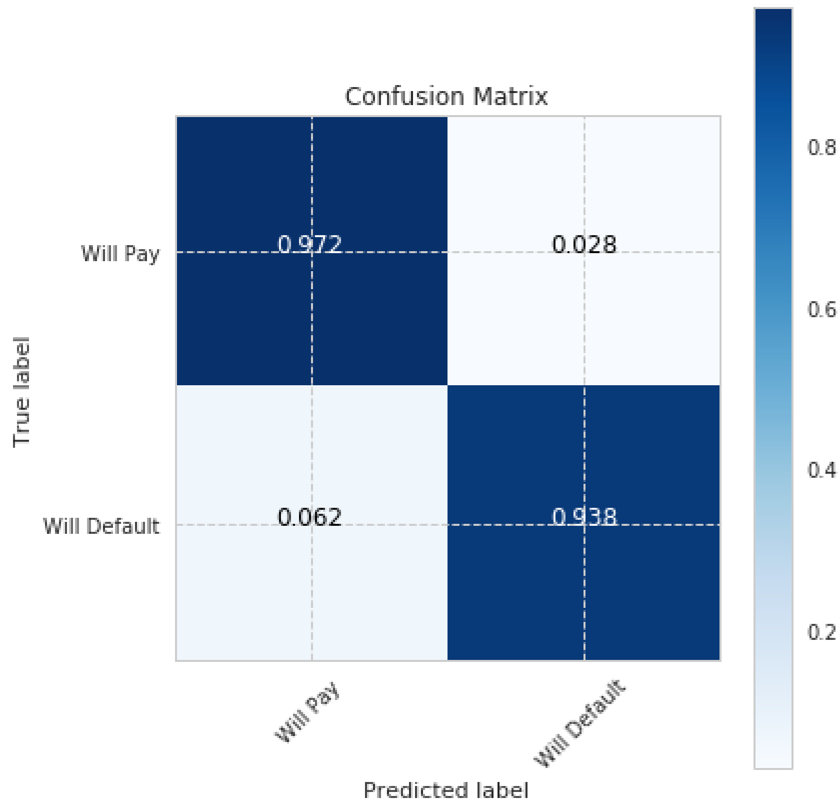
In [142]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.cross_validation import cross_val_score

myList = list(range(0,5))
neighbors = list(filter(lambda x: x % 2 != 0, myList))

# empty list that will hold cv scores
cv_scores = []

# perform 10-fold cross validation
for k in neighbors:
    knn = KNeighborsClassifier(n_neighbors=k)
    scores = cross_val_score(knn, X_train, y_train, cv=5, scoring='accuracy')
    cv_scores.append(scores.mean())

# changing to misclassification error
MSE = [1 - x for x in cv_scores]

# determining best k
optimal_k = neighbors[MSE.index(min(MSE))]
print('\nThe optimal number of neighbors is %d.' % optimal_k)

# plot misclassification error vs k
plt.plot(neighbors, MSE)
plt.xlabel('Number of Neighbors K')
plt.ylabel('Misclassification Error')
plt.show()
```

```
The optimal number of neighbors is 3.

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\font_manager.py:1316: Use
rWarning: findfont: Font family ['Ricty'] not found. Falling back to DejaVu San
s
  (prop.get_family(), self.defaultFamily[fontext]))
```


0.064
0.062
0.060
0.058
0.056
0.054
0.052
0.050
Misclassification Error
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00
Number of Neighbors K

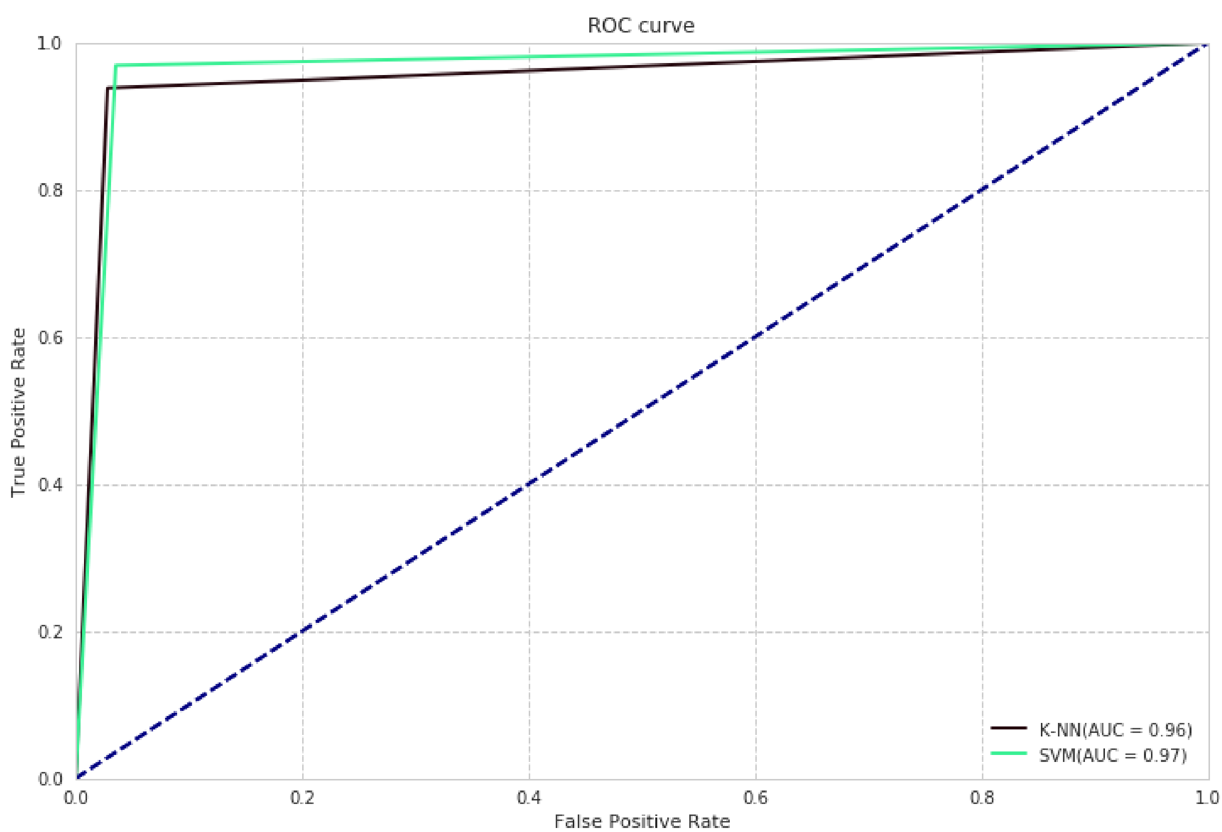
In [143]:
```
from sklearn.neighbors import KNeighborsClassifier
neigh = KNeighborsClassifier(n_neighbors=3)
neigh.fit(X_train,y_train)
LR_Predict_bin1 = neigh.predict(X_test)
LR_Accuracy = accuracy_score(y_test,LR_Predict_bin1.round())
print("KNN accuracy is ",LR_Accuracy)
plt.figure(figsize=(6,6))
plot_confusion_matrix(LR_Predict_bin1, normalize=True)
plt.show()
```

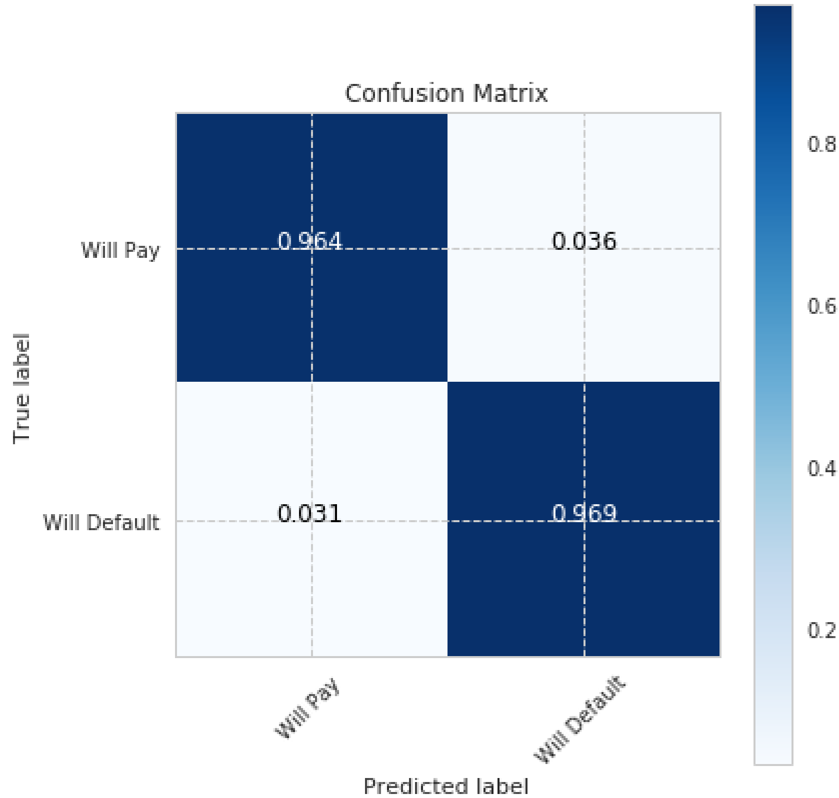
```
KNN accuracy is  0.954948301329

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\font_manager.py:1316: Use
rWarning: findfont: Font family ['Ricty'] not found. Falling back to DejaVu San
s
  (prop.get_family(), self.defaultFamily[fontext]))
```



In [147]:
```
from sklearn import svm
clf_svm = svm.SVC()
clf_svm.fit(X_train,y_train)
#LR_Predict1 = clf_svm.predict_proba(X_test)[:,1]
LR_Predict_bin2 = clf_svm.predict(X_test)
LR_Accuracy = accuracy_score(y_test,LR_Predict_bin2.round())
print("SVM accuracy is ",LR_Accuracy)

LR_Predict_train = clf_svm.predict(X_train)
LR_Accuracy1 = accuracy_score(y_train,LR_Predict_train.round())
print("SVM training accuracy is ",LR_Accuracy1)

plotAUC(y_test,LR_Predict_bin1,'K-NN')
plotAUC(y_test,LR_Predict_bin2,'SVM')
plt.show()
plt.figure(figsize=(6,6))
plot_confusion_matrix(LR_Predict_bin2, normalize=True)
plt.show()
```

```
SVM accuracy is  0.966765140325
SVM training accuracy is  0.990578237576

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\font_manager.py:1316: UserWarning: findfont: Font family ['Ricty'] not found. Falling back to DejaVu Sans
  (prop.get_family(), self.defaultFamily[fontext]))
```





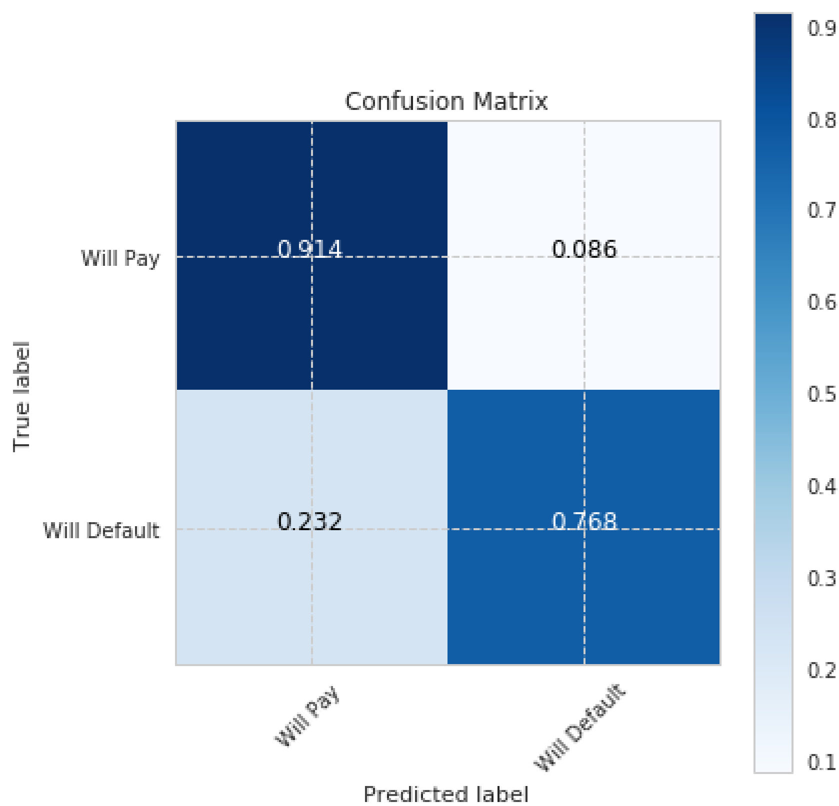
In [151]:

```
from sklearn.grid_search import RandomizedSearchCV
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier()
maxFeatures = range(1,data_clean.shape[1]-1)
param_dist = dict(max_features=maxFeatures)
rand = RandomizedSearchCV(rf, param_dist, cv=10, scoring='accuracy', n_iter=len(m
X=data_clean.iloc[:,:-1].values
y=data_clean.iloc[:,-1].values
rand.fit(X,y)
mean_scores = [result.mean_validation_score for result in rand.grid_scores_]
#print('Best Accuracy = '+str(rand.best_score_))
print(rand.best_estimator_)
```

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features=1, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

In [165]:
```
randomForest = RandomForestClassifier(bootstrap=True,criterion = "gini",max_featu

randomForest.fit(X_train,y_train)
rfPredict = randomForest.predict(X_test)
rfPredicttrain = randomForest.predict(X_train)
rfAccuracytrain = accuracy_score(y_train,rfPredicttrain)
rfPredictproba = randomForest.predict_proba(X_test)[:,1] #for ROC curve
rfAccuracy = accuracy_score(y_test,rfPredict)
roc_score = metrics.roc_auc_score(y_test,rfPredict)
print(rfAccuracy)
print(rfAccuracytrain)
plt.figure(figsize=(6,6))
plot_confusion_matrix(rfPredict, normalize=True)
plt.show()
```

```
0.84047267356
0.999630519121

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\font_manager.py:1316: Use
rWarning: findfont: Font family ['Ricty'] not found. Falling back to DejaVu San
s
  (prop.get_family(), self.defaultFamily[fontext]))
```



In [140]:
```
from sklearn.model_selection import cross_val_score
import numpy as np
X=data_clean.iloc[:,:-1].values
y=data_clean.iloc[:,-1].values
clf = RandomForestClassifier()
print(np.mean(cross_val_score(clf, X_train, y_train, cv=15)))

param_grid = {
                 'n_estimators': [5, 10, 15, 20],
                 'max_depth': [2, 5, 6, 7,8,9]
             }
from sklearn.grid_search import GridSearchCV

grid_clf = GridSearchCV(clf, param_grid, cv=15)
grid_clf.fit(X_train, y_train)
grid_clf. best_estimator_
grid_clf. best_params_
grid_clf.grid_scores_
```

```
0.906330308745
```

Out[140]:
```
[mean: 0.72178, std: 0.04284, params: {'max_depth': 2, 'n_estimators': 5},
 mean: 0.75559, std: 0.02661, params: {'max_depth': 2, 'n_estimators': 10},
 mean: 0.76575, std: 0.02620, params: {'max_depth': 2, 'n_estimators': 15},
 mean: 0.77813, std: 0.02065, params: {'max_depth': 2, 'n_estimators': 20},
 mean: 0.81877, std: 0.02123, params: {'max_depth': 5, 'n_estimators': 5},
 mean: 0.83244, std: 0.01893, params: {'max_depth': 5, 'n_estimators': 10},
 mean: 0.84075, std: 0.01509, params: {'max_depth': 5, 'n_estimators': 15},
 mean: 0.84463, std: 0.01556, params: {'max_depth': 5, 'n_estimators': 20},
 mean: 0.84075, std: 0.01172, params: {'max_depth': 6, 'n_estimators': 5},
 mean: 0.85849, std: 0.01110, params: {'max_depth': 6, 'n_estimators': 10},
 mean: 0.85960, std: 0.01494, params: {'max_depth': 6, 'n_estimators': 15},
 mean: 0.86717, std: 0.01290, params: {'max_depth': 6, 'n_estimators': 20},
 mean: 0.86255, std: 0.01231, params: {'max_depth': 7, 'n_estimators': 5},
 mean: 0.87087, std: 0.01620, params: {'max_depth': 7, 'n_estimators': 10},
 mean: 0.87604, std: 0.01555, params: {'max_depth': 7, 'n_estimators': 15},
 mean: 0.88417, std: 0.01753, params: {'max_depth': 7, 'n_estimators': 20},
 mean: 0.87068, std: 0.01499, params: {'max_depth': 8, 'n_estimators': 5},
 mean: 0.88177, std: 0.01700, params: {'max_depth': 8, 'n_estimators': 10},
 mean: 0.89174, std: 0.01349, params: {'max_depth': 8, 'n_estimators': 15},
 mean: 0.89248, std: 0.01620, params: {'max_depth': 8, 'n_estimators': 20},
 mean: 0.87308, std: 0.01432, params: {'max_depth': 9, 'n_estimators': 5},
 mean: 0.89710, std: 0.01639, params: {'max_depth': 9, 'n_estimators': 10},
 mean: 0.89987, std: 0.01253, params: {'max_depth': 9, 'n_estimators': 15},
 mean: 0.90227, std: 0.01516, params: {'max_depth': 9, 'n_estimators': 20}]
```