In [11]:
```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

In [12]:
```python
df=pd.read_csv(r'C:\Users\sharma17\Downloads\ml\classification dataset\loan_data.csv')
```

In [13]:
```python
df.head()
```

Out[13]:

	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	not.fully.paid
0	1	debt_consolidation	0.1189	829.10	11.350407	19.48	737	5639.958333	28854	52.1	0	0	0	0
1	1	credit_card	0.1071	228.22	11.082143	14.29	707	2760.000000	33623	76.7	0	0	0	0
2	1	debt_consolidation	0.1357	366.86	10.373491	11.63	682	4710.000000	3511	25.6	1	0	0	0
3	1	debt_consolidation	0.1008	162.34	11.350407	8.10	712	2699.958333	33667	73.2	1	0	0	0
4	1	credit_card	0.1426	102.92	11.299732	14.97	667	4066.000000	4740	39.5	0	1	0	0

In [14]:
```python
df['paid']=df['not.fully.paid']
```

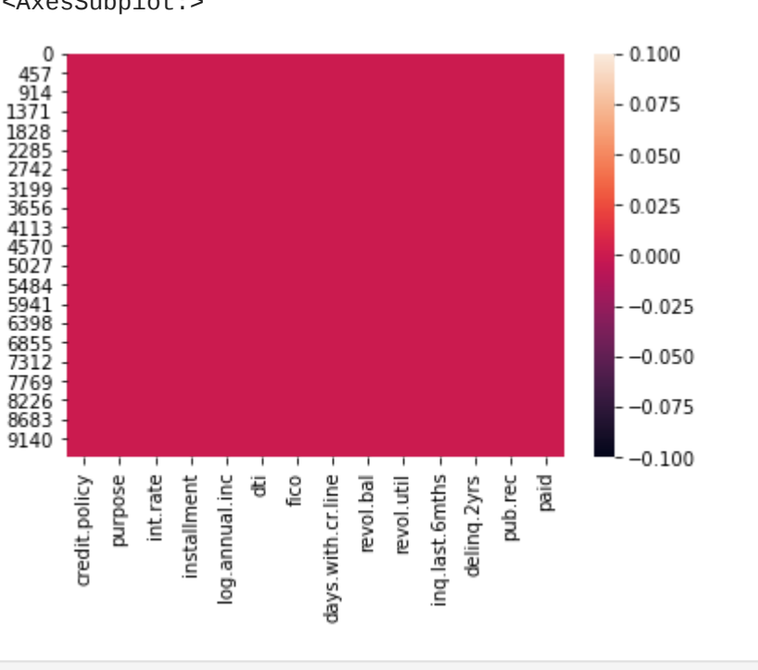
In [15]:
```python
df.drop('not.fully.paid',inplace=True,axis=1)
```

In [21]:
```python
df_cat=df.purpose
from sklearn.preprocessing import LabelEncoder as le
```

In [22]:
```python
l=le()
df_cat=l.fit_transform(df_cat)
```

In [37]:
```python
sns.heatmap(df.isnull())
plt.show()
```

Out[37]: <AxesSubplot:>

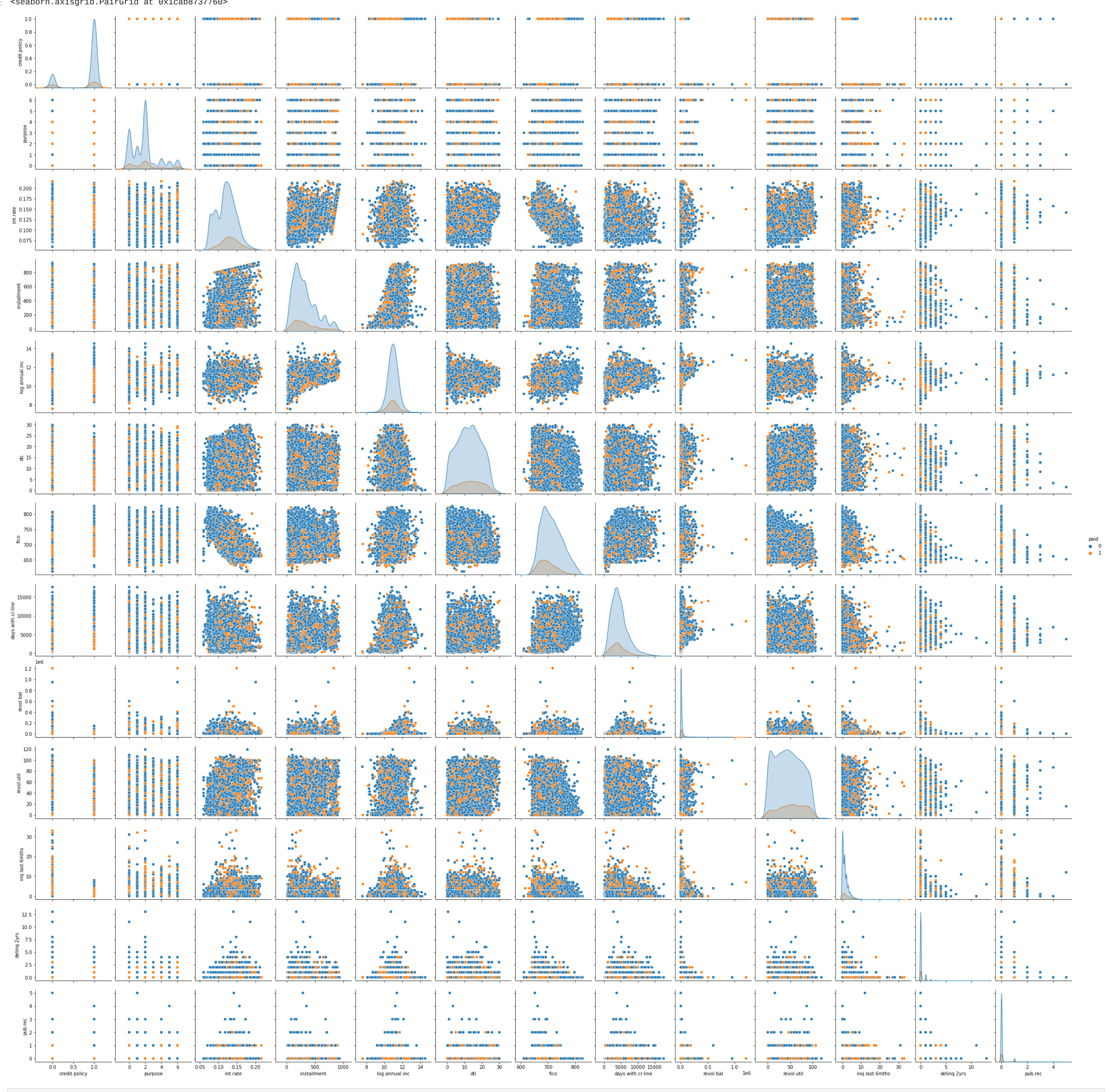


In [40]:
```python
df.paid.value_counts()
```

Out[40]:
```
0    8045
1    1533
Name: paid, dtype: int64
```

In [36]:
```python
sns.pairplot(df,hue='paid')
```

Out[36]: <seaborn.axisgrid.PairGrid at 0x1cab8737760>



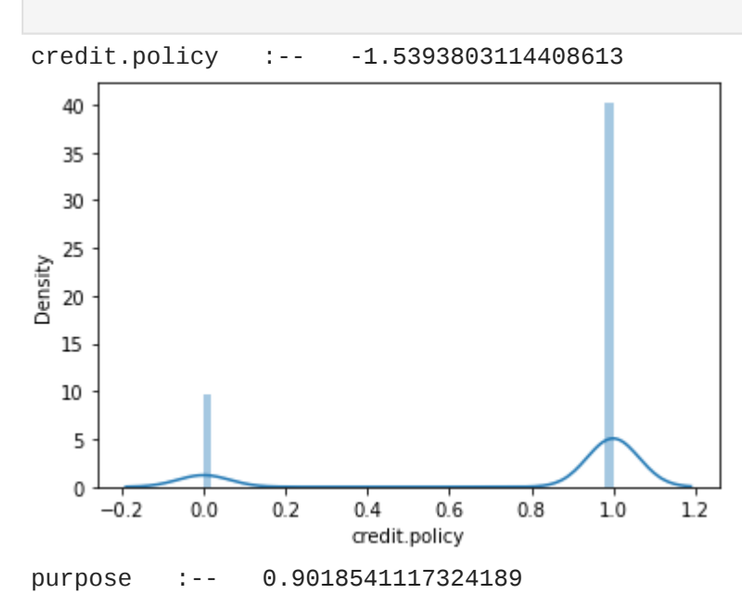
In [42]:
```python
plt.figure(figsize=(15,12))
sns.heatmap(df.corr(),annot=True)
plt.show()
```

In [45]:
```python
from scipy.stats import skew
```

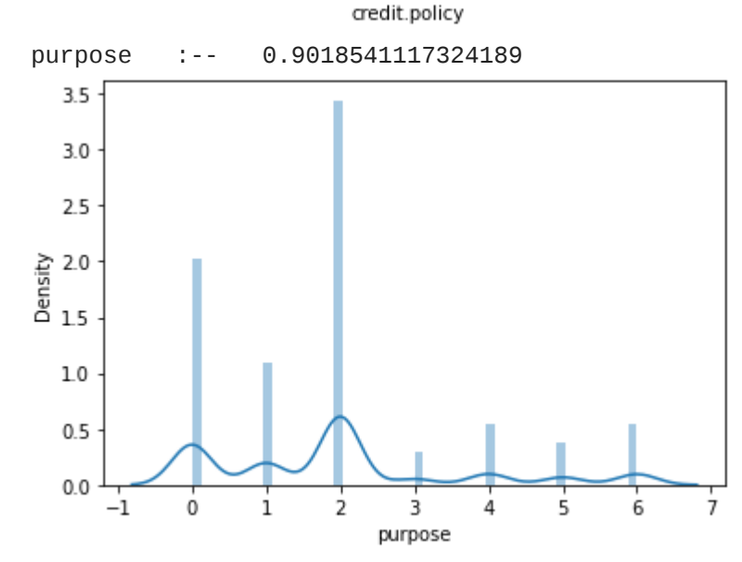
In [51]:
```python
for col in df:

    print (col,'  :--   ',end='' )
    print(skew(df[col]))
    sns.distplot(df[col])
    plt.show()
```

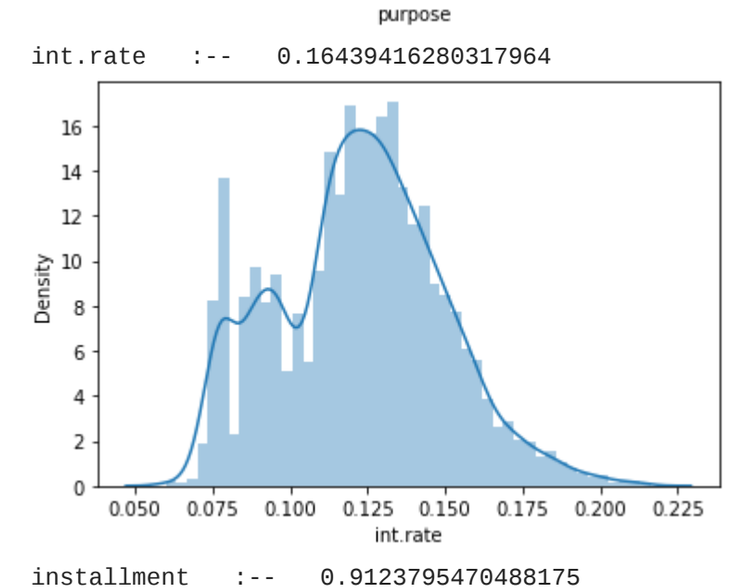
credit.policy   :--   -1.5393803114408613



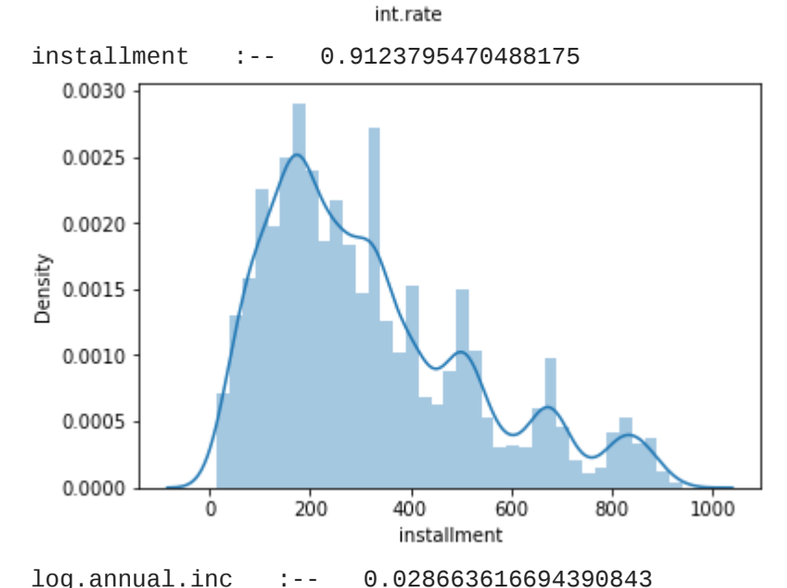
purpose   :--   0.9018541117324189



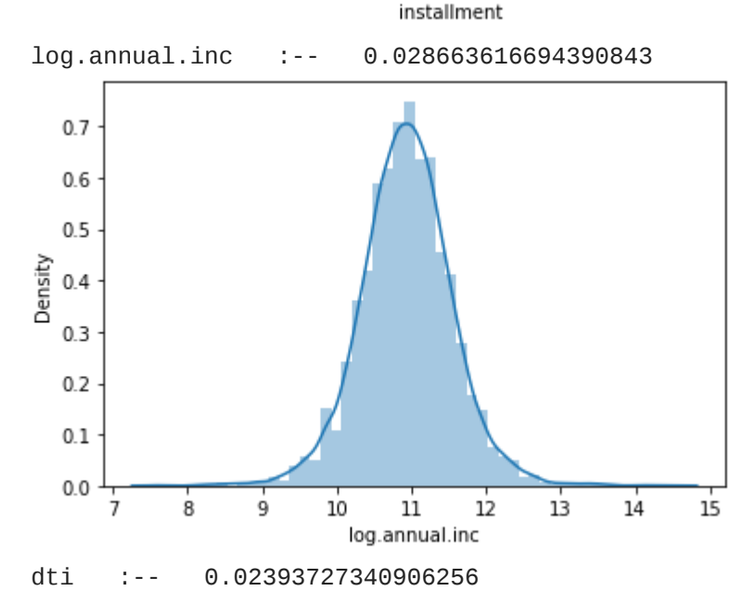
int.rate   :--   0.16439416280317964



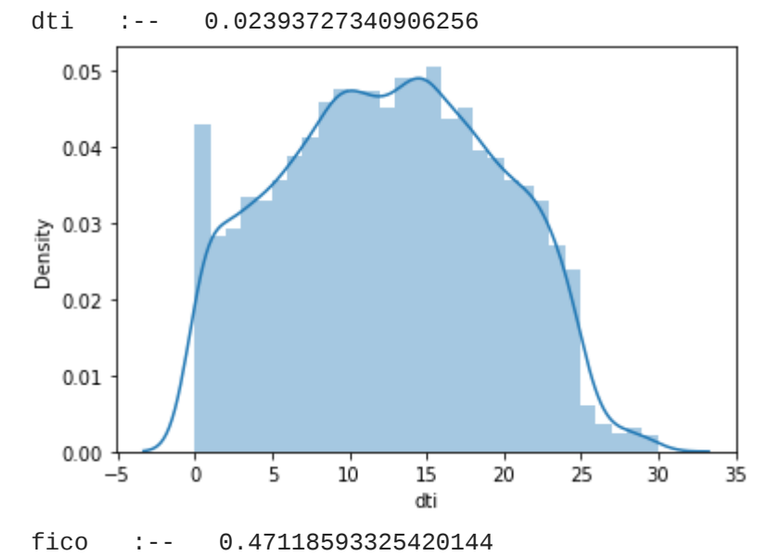
installment   :--   0.9123795470488175



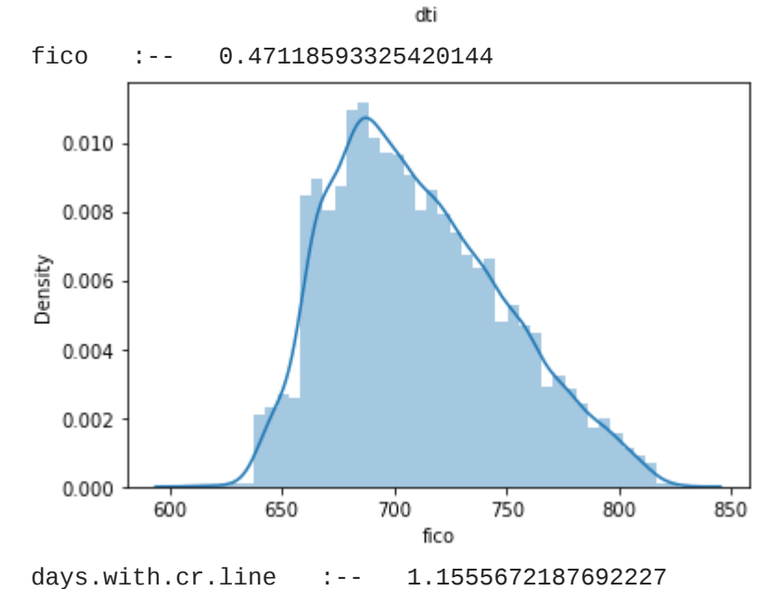
log.annual.inc   :--   0.028663616694390843



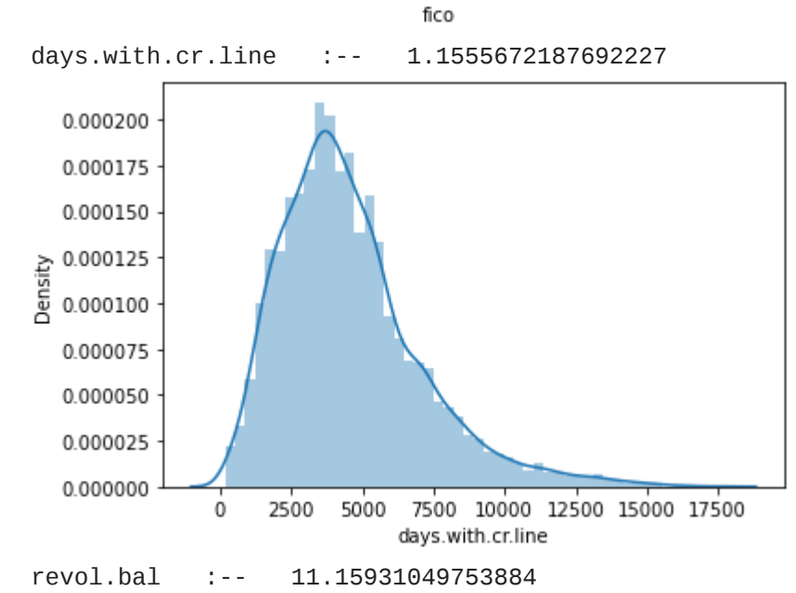
dti   :--   0.023937273409962 56



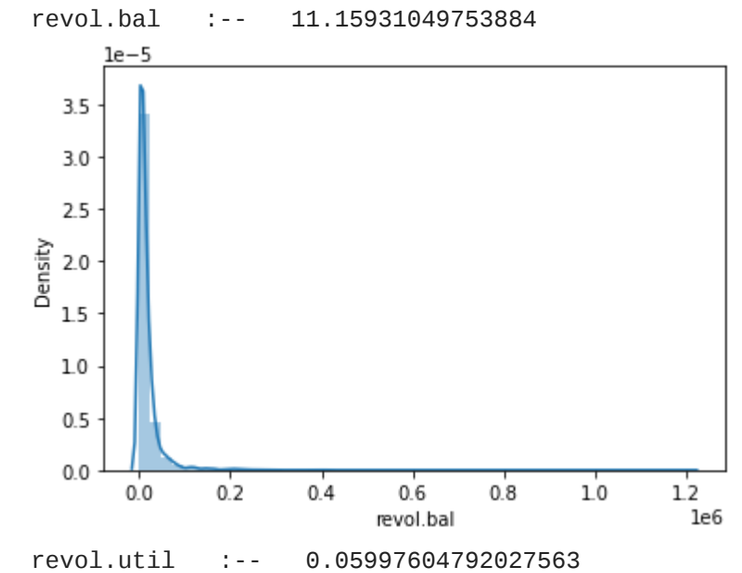
fico   :--   0.47118593325420144



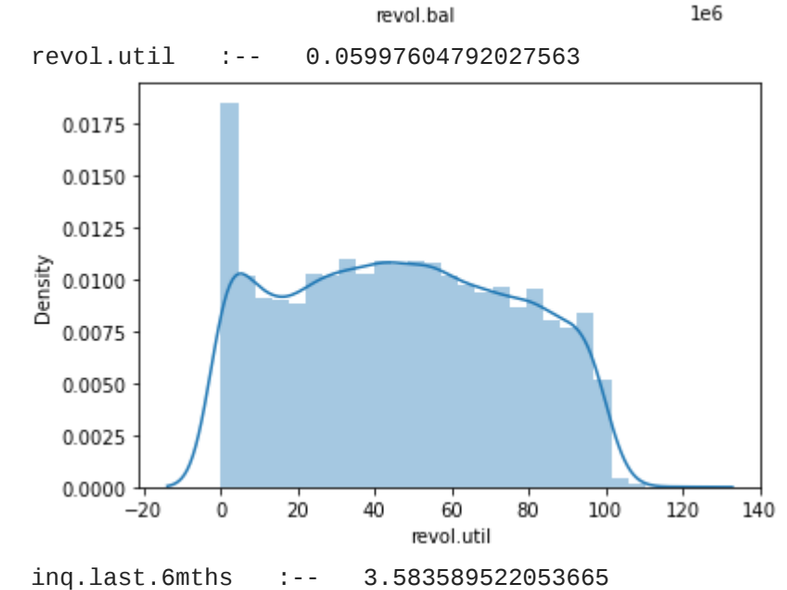
days.with.cr.line   :--   1.1555672187692227



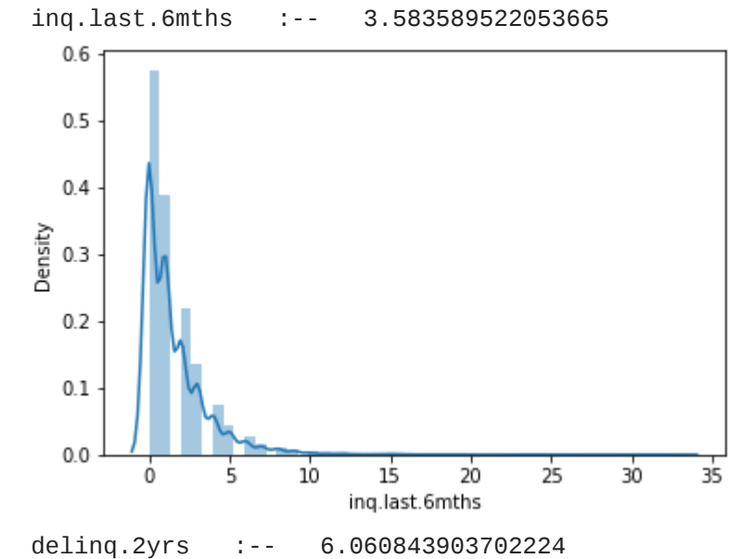
revol.bal   :--   11.15931049753884



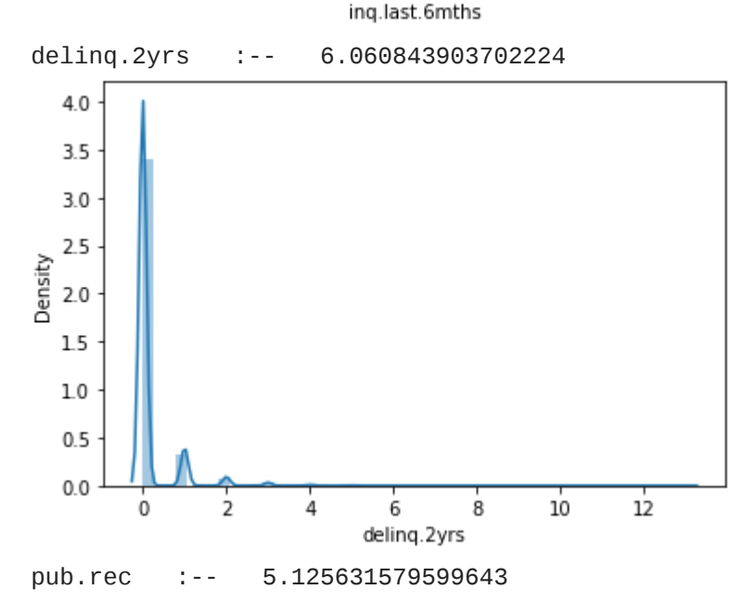
revol.util   :--   0.05997604792027563



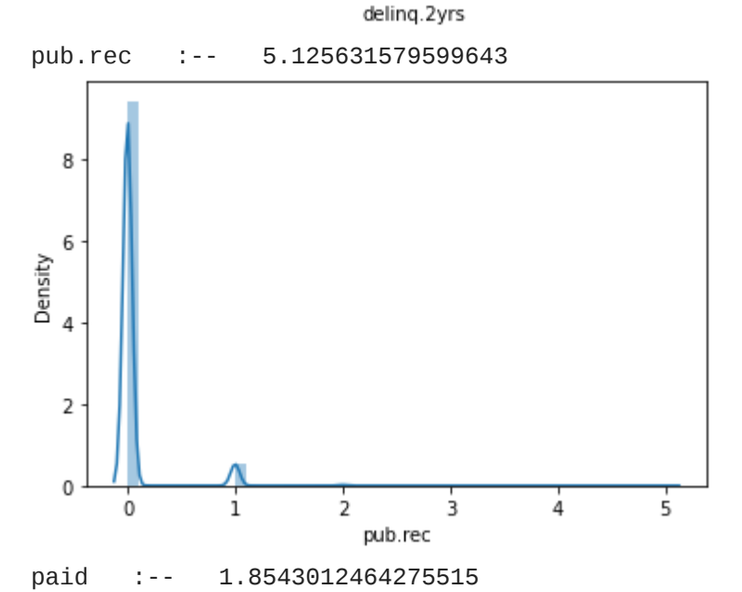
inq.last.6mths   :--   3.583589522053665



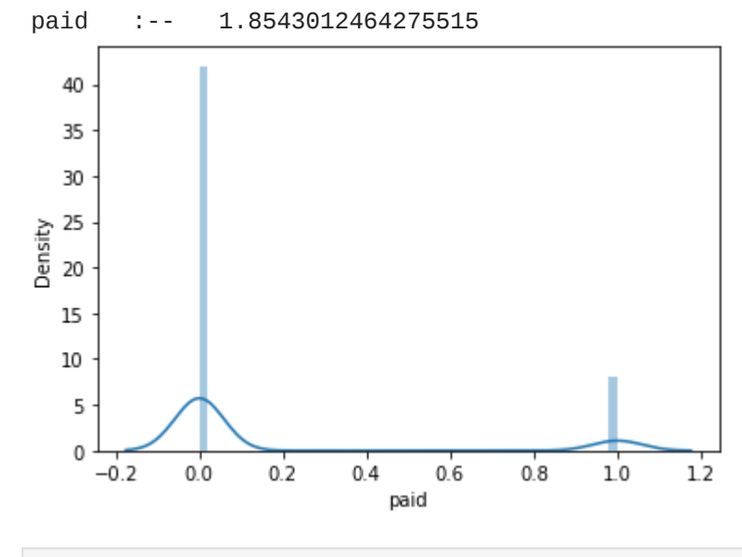
delinq.2yrs   :--   6.060843903702224



pub.rec   :--   5.125631579599643



paid   :--   1.8543012464275515



In [78]:
```python
df.purpose=df_cat
x=df.iloc[:,:-1]
y=df.iloc[:,-1]
```

In [79]:
```python
from sklearn.preprocessing import StandardScaler as sc
```

In [80]:
```python
s=sc()
s.fit(x)
x=s.transform(x)
```

In [165…
```python
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier ,AdaBoostClassifier,GradientBoostingClassifier
from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split ,StratifiedKFold,cross_val_score
from sklearn.metrics import accuracy_score ,confusion_matrix ,classification_report
```

In [156…
```python
models=[('Logistic Regression',LogisticRegression()),
        ('Decision Tree',DecisionTreeClassifier()),
        ('Support Vector ',SVC()),
        ('Nearest Neighbors',KNeighborsClassifier()),
        ('Random Forest ',RandomForestClassifier()),
        ('AdaBoostClassifier',AdaBoostClassifier()),
        ('Gradient Boosting',GradientBoostingClassifier()),
        ('Xtreame Gradient Boosting',XGBClassifier())

]
```

In [157…
```python
xtrain,xtest,ytrain,ytest=train_test_split(x,y,random_state=0,test_size=0.30)
```

In [158…
```python
accuracy=[]

for name,model in models:

    print(name,'  :--  ')
    print()
    model.fit(xtrain,ytrain)
    print()
    ypred=model.predict(xtest)
    print(confusion_matrix(ytest,ypred))
    print()

    print(classification_report(ytest,ypred))
    accuracy.append(accuracy_score(ytest,ypred))
```

```
Logistic Regression   :--


[[2392   14]
 [ 465    3]]

              precision    recall  f1-score   support

           0       0.84      0.99      0.91      2406
           1       0.18      0.01      0.01       468

    accuracy                           0.83      2874
   macro avg       0.51      0.50      0.46      2874
weighted avg       0.73      0.83      0.76      2874

Decision Tree   :--


[[2007  399]
 [ 354  114]]

              precision    recall  f1-score   support

           0       0.85      0.83      0.84      2406
           1       0.22      0.24      0.23       468

    accuracy                           0.74      2874
   macro avg       0.54      0.54      0.54      2874
weighted avg       0.75      0.74      0.74      2874

Support Vector    :--


[[2404    2]
 [ 466    2]]

              precision    recall  f1-score   support

           0       0.84      1.00      0.91      2406
           1       0.50      0.00      0.01       468

    accuracy                           0.84      2874
   macro avg       0.67      0.50      0.46      2874
weighted avg       0.78      0.84      0.76      2874

Nearest Neighbors   :--


[[2324   82]
 [ 441   27]]

              precision    recall  f1-score   support

           0       0.84      0.97      0.90      2406
           1       0.25      0.06      0.09       468

    accuracy                           0.82      2874
   macro avg       0.54      0.51      0.50      2874
weighted avg       0.74      0.82      0.77      2874

Random Forest    :--


[[2386   20]
 [ 456   12]]

              precision    recall  f1-score   support

           0       0.84      0.99      0.91      2406
           1       0.38      0.03      0.05       468

    accuracy                           0.83      2874
   macro avg       0.61      0.51      0.48      2874
weighted avg       0.76      0.83      0.77      2874

AdaBoostClassifier   :--


[[2388   18]
 [ 455   13]]

              precision    recall  f1-score   support

           0       0.84      0.99      0.91      2406
           1       0.42      0.03      0.05       468

    accuracy                           0.84      2874
   macro avg       0.63      0.51      0.48      2874
weighted avg       0.77      0.84      0.77      2874

Gradient Boosting   :--


[[2386   20]
 [ 455   13]]

              precision    recall  f1-score   support

           0       0.84      0.99      0.91      2406
           1       0.39      0.03      0.05       468

    accuracy                           0.83      2874
   macro avg       0.62      0.51      0.48      2874
weighted avg       0.77      0.83      0.77      2874

Xtreame Gradient Boosting   :--

[19:02:10] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
[[2330   76]
 [ 428   40]]

              precision    recall  f1-score   support

           0       0.84      0.97      0.90      2406
           1       0.34      0.09      0.14       468

    accuracy                           0.82      2874
   macro avg       0.59      0.53      0.52      2874
weighted avg       0.76      0.82      0.78      2874
```

In [160…
```python
print('the mean accuracy of the model  :--  ',np.array(accuracy).mean()*100)
```

```
the mean accuracy of the model  :--   81.94589422407795
```

In [143…
```python
xgb=XGBClassifier(learning_rate=16.9,gamma=20)
xgb.fit(xtrain,ytrain)
ypred=xgb.predict(xtest)
print('trainig score  :-  ',xgb.score(xtrain,ytrain)*100)
print('testing score   :-  ',xgb.score(xtest,ytest)*100)
```

```
[18:56:06] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
trainig score  :-   84.11396181384248
testing score   :-   83.7160751565762
```

In [147…
```python
skf=StratifiedKFold(n_splits=5)
skf.get_n_splits(x,y)
```

Out[147… 5

In [161…
```python
accuracy=[]

for train,test in skf.split(x,y):
    xtrain,xtest=x[train],x[test]
    ytrain,ytest=y[train],y[test]


    xgb=XGBClassifier(learning_rate=16.9,gamma=20)
    xgb.fit(xtrain,ytrain)
    ypred=xgb.predict(xtest)
    accuracy.append(accuracy_score(ytest,ypred))
```

```
[19:03:02] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
[19:03:02] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
[19:03:02] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
[19:03:03] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
[19:03:03] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
```

Because the dataset is imbalance we are getting less accuracy

In [173…
```python
print('this is the  minimum accuracy we can get from xtream boosting classifier  :--',np.array(accuracy).mean()*100)
```

```
this is the  minimum accuracy we can get from xtream boosting classifier  :-- 73.10162599410215
```

In [ ]: