Micro Credit Loan

Submitted by:

Mahaboob Basha Shaik

ACKNOWLEDGMENT

I would like to express my special thanks of gratitude to my SME Mr. Shubham Yadav as well Fliprobo who gave me the golden opportunity to do this wonderful project on the Micro Credit Loan Project, which also helped me in doing a lot of Research and i came to know about so many new things I am thankful to them.

Secondly, I would also like to thank my parents and friends who helped me a lot in finalizing this project within the limited time frame.

INTRODUCTION

- Business Problem Framing

  Identify the target customers who are credible to payback the loans which can be identified with their usage of their mobile phone recharges and monthly balance maintenance.

- Conceptual Background of the Domain Problem

  Communication has become important factor in everyone day to day life. MFI wants to collaborate with telecom industry to reach out all poor customers and low-income families who can use this opportunity and payback the sanctioned amount in stipulated time frame.

- Review of Literature

  we study and discuss the MFI and Telecom Credit strategies in a telecommunication industry. The goal of this study is to apply various analysis techniques along with data mining techniques to predict customer credibility in repay of loans taken from a telecommunications company. The study will help telecommunications companies to understand customer repayable capacity.

- Motivation for the Problem Undertaken

  MFI working to Telecom Industry to provide loans to low level income which help in not happening their connectivity with their near and dear once.

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

  Linear Regression is applied for the project completion. In which I used Lasso Regressor, Ridge Regressor, Decision Tree, Random Forest Regressor and GridsearchCV for finding the best model.

- Data Sources and their formats

	int64	object	float64
Count	13	3	21

```
: Unnamed: 0                int64
  label                     int64
  msisdn                   object
  aon                     float64
  daily_decr30            float64
  daily_decr90            float64
  rental30                float64
  rental90                float64
  last_rech_date_ma       float64
  last_rech_date_da       float64
  last_rech_amt_ma          int64
  cnt_ma_rech30             int64
  fr_ma_rech30            float64
  sumamnt_ma_rech30       float64
  medianamnt_ma_rech30    float64
  medianmarechprebal30    float64
  cnt_ma_rech90             int64
  fr_ma_rech90              int64
  sumamnt_ma_rech90         int64
  medianamnt_ma_rech90    float64
  medianmarechprebal90    float64
  cnt_da_rech30           float64
  fr_da_rech30            float64
  cnt_da_rech90             int64
  fr_da_rech90              int64
  cnt_loans30               int64
  amnt_loans30              int64
  maxamnt_loans30         float64
  medianamnt_loans30      float64
  cnt_loans90             float64
  amnt_loans90              int64
  maxamnt_loans90           int64
  medianamnt_loans90      float64
  payback30               float64
  payback90               float64
  pcircle                  object
  pdate                    object
  dtype: object
```

- Data Pre-processing Done

  Verifying the dataset weather it is having NULL values and checking the data types of the provided data. Converted all the float data type to Int data type for modelling and removing the skewness / outliers from computing.

- Data Inputs- Logic- Output Relationships

  We had considered y variable as Label as it explains weather the customer had made the repayment of loan. X variable consist of other variables which consist of Avg of 30 days and 90days of parameters which directly or indirectly influences the y variable.

- Hardware and Software Requirements and Tools Used

  a. import numpy as np
  b. import pandas as pd
  c. import sklearn
  d. import seaborn as sb
  e. import matplotlib.pyplot as plt
  f. from scipy.stats import zscore
  g. import numpy as np
  h. from sklearn.preprocessing import StandardScaler
  i. from sklearn.preprocessing import power_transform
  j. from sklearn.model_selection import train_test_split
  k. from sklearn.linear_model import LinearRegression
  l. from sklearn.metrics import r2_score
  m. from sklearn.model_selection import GridSearchCV
  n. from sklearn.model_selection import cross_val_score
  o. from sklearn.linear_model import Ridge
  p. from sklearn.tree import DecisionTreeRegressor
  q. from sklearn.ensemble import RandomForestRegressor
  r. from sklearn.ensemble import GradientBoostingRegressor
  s. from sklearn.svm import SVR

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

  Decision Tree is the best possible model in solving the model as it has cross validation and r2 score of 100. All data points are interacting with the predicted model.

- Testing of Identified Approaches (Algorithms)

  Lasso Regressor, Ridge Regressor, Decision Tree Regressor, Random Forest Regressor, GridsearchCV

- Run and Evaluate selected models

Regularlisation - Lasso

```
from sklearn.linear_model import Lasso
parameters ={'alpha':[.00001,.0001,.001,.01,.1,1,10],'random_state':list(range(0,10))}
ls=Lasso()
clf=GridSearchCV(ls,parameters)
clf.fit(x_train,y_train)
print (clf.best_params_)
```

```
{'alpha': 1e-05, 'random_state': 0}
```

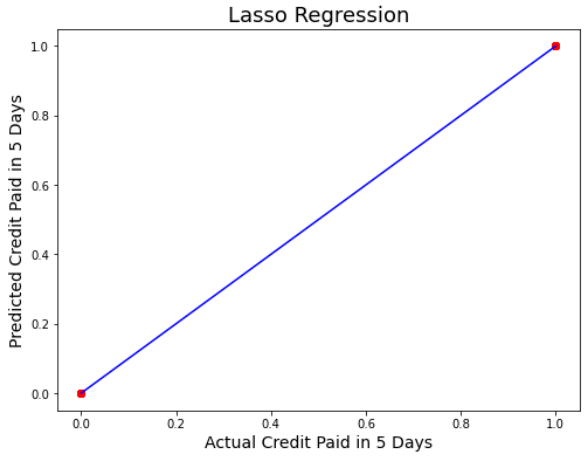
```
ls=Lasso(alpha=0.0001,random_state=0)
ls.fit(x_train,y_train)
ls.score(x_train,y_train)
pred_ls=ls.predict(x_test)

lss=r2_score(y_test,pred_ls)
for j in range(2,10):
    lsscore = cross_val_score(ls,x_t,y,cv=j)
    lsc=lsscore.mean()
    print("At CV :-",j)
    print("Cross Validation Score is :-",lsc*100)
    print ("R2_score is :-",lss*100)
    print('\n')
```

```
At CV :- 2
Cross Validation Score is :- 99.99999084559663
R2_score is :- 99.99999083206609


At CV :- 3
Cross Validation Score is :- 99.99999084584479
R2_score is :- 99.99999083206609
```

```
from sklearn.metrics import mean_squared_error,mean_absolute_error
print("Error:")
print("Mean Absolute Error:",mean_absolute_error(y_test,pred_ls))
print("Mean Square Error:",mean_squared_error(y_test,pred_ls))
print("Root Mean Sqaured Error:", np.sqrt(mean_squared_error(y_test,pred_ls)))
```

```
Error:
Mean Absolute Error: 6.62293035516008e-05
Mean Square Error: 1.0045669535635966e-08
Root Mean Sqaured Error: 0.0001002280875585081
```

```
plt.figure(figsize=(8,6))
plt.scatter(x=y_test,y=pred_ls,color='r')
plt.plot(y_test,y_test,color='b')
plt.xlabel('Actual Credit Paid in 5 Days',fontsize=14)
plt.ylabel('Predicted Credit Paid in 5 Days',fontsize=14)
plt.title('Lasso Regression',fontsize=18)
plt.show()
```



Data Point is intracting with the predicted line

Ridge Regression

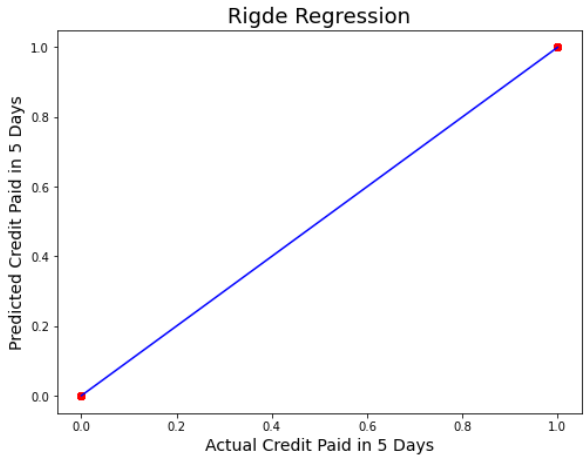
```
from sklearn.linear_model import Ridge
parameters ={'alpha':[.0001,.001,.01,.1,1],'fit_intercept':[True,False],'normalize':[True,False],'random_state':[1,2,3,4,5,6
rd=Ridge()
clf=GridSearchCV(rd,parameters)
clf.fit(x_train,y_train)
print (clf.best_params_)
```

```
{'alpha': 0.0001, 'fit_intercept': True, 'normalize': False, 'random_state': 1}
```

```
ridge=Ridge(alpha=0.0001,random_state=1,fit_intercept=True,normalize= False)
rd.fit(x_train,y_train)
rd.score(x_train,y_train)
pred_rd=rd.predict(x_test)

rdd=r2_score(y_test,pred_rd)
for j in range(2,10):
    rdscore = cross_val_score(rd,x_t,y,cv=j)
    rdc=rdscore.mean()
    print("At CV :-",j)
    print("Cross Validation Score is :-",rdc*100)
    print ("R2_score is :-",rdc*100)
    print('\n')
```

```
At CV :- 2
Cross Validation Score is :- 99.99999998983795
R2_score is :- 99.99999998983795


At CV :- 3
Cross Validation Score is :- 99.99999999428626
R2_score is :- 99.99999999428626


At CV :- 4
Cross Validation Score is :- 99.9999999954857
R2_score is :- 99.9999999954857
```

```
plt.figure(figsize=(8,6))
plt.scatter(x=y_test,y=pred_rd,color='r')
plt.plot(y_test,y_test,color='b')
plt.xlabel('Actual Credit Paid in 5 Days',fontsize=14)
plt.ylabel('Predicted Credit Paid in 5 Days',fontsize=14)
plt.title('Rigde Regression',fontsize=18)
plt.show()
```



Same as Lasso Model all Data points are interacting with the predictd line

Ensemblier Technique - Decision Tree

```
from sklearn.tree import DecisionTreeRegressor
parameters={'criterion':['mse','friedman_mse','mse'],'splitter':['best','random']}
dt=DecisionTreeRegressor()
clf=GridSearchCV(dt,parameters)
clf.fit(x_train,y_train)
print (clf.best_params_)
```

```
{'criterion': 'mse', 'splitter': 'best'}
```

```
dt=DecisionTreeRegressor(criterion='mse', splitter='best')
dt.fit(x_train,y_train)
dt.score(x_train,y_train)
pred_decision=dt.predict(x_test)
dts=r2_score(y_test,pred_decision)
print("r2_score:",dts*100)
dtscore = cross_val_score(dt,x_t,y,cv=3)
dtc=dtscore.mean()
print('Cross Val Score:',dtc*100)
```

```
r2_score: 100.0
Cross Val Score: 100.0
```

```
print("Error:")
print("Mean Absolute Error:",round(mean_absolute_error(y_test,pred_decision),2))
print("Mean Square Error:",round(mean_squared_error(y_test,pred_decision),2))
print("Root Mean Sqaured Error:",round(np.sqrt(mean_squared_error(y_test,pred_decision)),2))
```

```
Error:
Mean Absolute Error: 0.0
Mean Square Error: 0.0
Root Mean Sqaured Error: 0.0
```

```
plt.figure(figsize=(8,6))
plt.scatter(x=y_test,y=pred_decision,color='r')
plt.plot(y_test,y_test,color='b')
plt.xlabel('Actual Credit Paid in 5 Days',fontsize=14)
plt.ylabel('Predicted Credit Paid in 5 Days',fontsize=14)
plt.title('Decision Tree Regression',fontsize=18)
plt.show()
```

R2 score and CV Score are 100 , All The Points are interacting with predicted line

Random Forest

```
from sklearn.ensemble import RandomForestRegressor
parameters={'criterion':['mse','friedman_mse','mse'],'n_estimators':[100,200,300]}
rf=RandomForestRegressor()
clf=GridSearchCV(rf,parameters)
clf.fit(x_train,y_train)
print (clf.best_params_)
```

```
{'criterion': 'mse', 'n_estimators': 100}
```

```
rf=RandomForestRegressor(criterion='mse', n_estimators= 100)
rf.fit(x_train,y_train)
rf.score(x_train,y_train)
pred_rd=rf.predict(x_test)
rfs=r2_score(y_test,pred_rd)
print("r2_score:",rfs*100)
rfscore = cross_val_score(rf,x_t,y,cv=3)
rfc=rfscore.mean()
print('Cross Val Score:',rfc*100)
```

```
r2_score: 100.0
Cross Val Score: 100.0
```

```
plt.figure(figsize=(8,6))
plt.scatter(x=y_test,y=pred_rd,color='r')
plt.plot(y_test,y_test,color='b')
plt.xlabel('Actual Credit Paid in 5 Days',fontsize=14)
plt.ylabel('Predicted Credit Paid in 5 Days',fontsize=14)
plt.title('Random Forest Regressor',fontsize=18)
plt.show()
```

In Random Forest Regressor also Data points are intracting with the predicted lines

Gradient Bosster Regressor

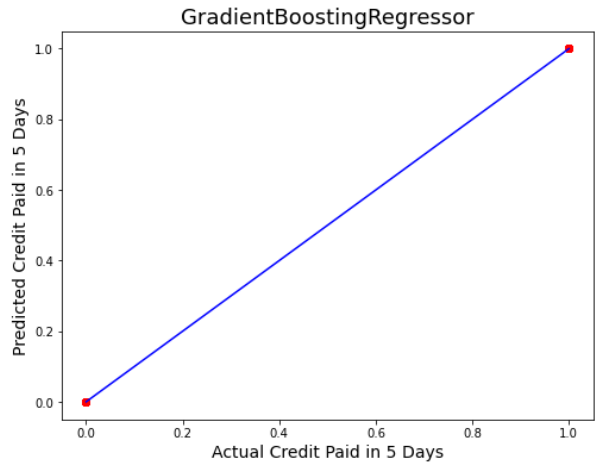
```
from sklearn.ensemble import GradientBoostingRegressor
parameters={'loss':['ls','lad','huber','quantile'],'n_estimators':[50,100,200],'criterion':['friedman_mse','mse']}
gbr=GradientBoostingRegressor()
clf=GridSearchCV(gbr,parameters)
clf.fit(x_train,y_train)
print(clf.best_params_)

```

```
{'criterion': 'friedman_mse', 'loss': 'ls', 'n_estimators': 200}
```

```
gbr=GradientBoostingRegressor(criterion='friedman_mse', loss='ls',n_estimators=100)
gbr.fit(x_train,y_train)
gbr.score(x_train,y_train)
pred_random=gbr.predict(x_test)
gbrs=r2_score(y_test,pred_random)
print("r2_score:",round(gbrs*100,2))
gbscore = cross_val_score(gbr,x_t,y,cv=3)
gbrc=gbscore.mean()
print('Cross Val Score:',round(gbrc*100,2))

```

```
r2_score: 100.0
Cross Val Score: 100.0
```

```
plt.figure(figsize=(8,6))
plt.scatter(x=y_test,y=pred_random,color='r')
plt.plot(y_test,y_test,color='b')
plt.xlabel('Actual Credit Paid in 5 Days',fontsize=14)
plt.ylabel('Predicted Credit Paid in 5 Days',fontsize=14)
plt.title('GradientBoostingRegressor',fontsize=18)
plt.show()
```



All Data points in this model also interseting with Predicted Line

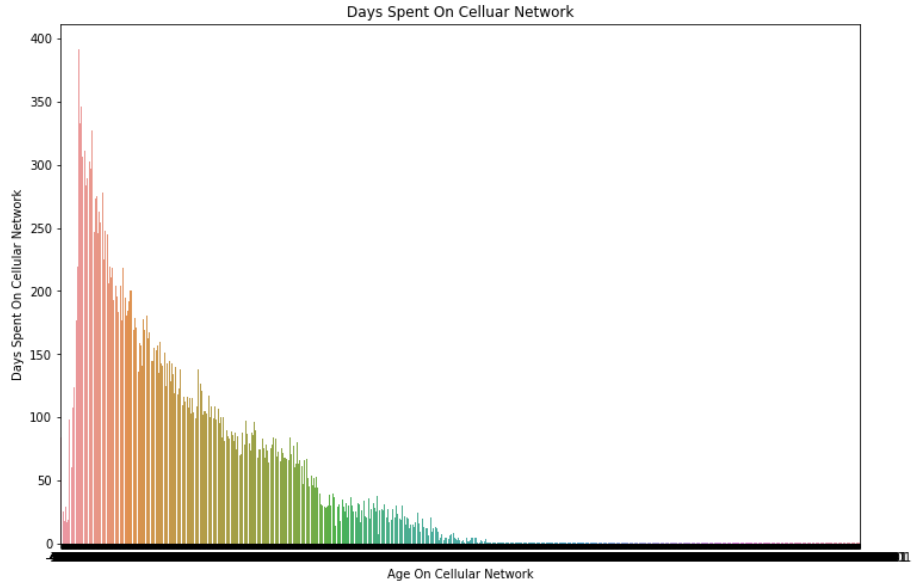
- Key Metrics for success in solving problem under consideration.

  Data type conversion, Elimation of outliers and skewed data from dataset.

- Visualizations

Uni Variant Ananlysis

```
:  1 plt.subplots(figsize=(12,8))
   2 sb.countplot(x='aon',data=df)
   3 plt.title("Days Spent On Celluar Network")
   4 plt.xlabel('Age On Cellular Network')
   5 plt.ylabel('Days Spent On Cellular Network')
   6 plt.show()
```



Highest days spent on network is between bin is in 350 - 400

- Interpretation of the Results

After all models in Linear Regression, I found Decision Tree Regression provide accurate prediction for the test data. Please find the below:

Ensemblier Technique - Decision Tree

```
from sklearn.tree import DecisionTreeRegressor
parameters={'criterion':['mse','friedman_mse','mse'],'splitter':['best','random']}
dt=DecisionTreeRegressor()
clf=GridSearchCV(dt,parameters)
clf.fit(x_train,y_train)
print (clf.best_params_)
```

```
{'criterion': 'mse', 'splitter': 'best'}
```

```
dt=DecisionTreeRegressor(criterion='mse', splitter='best')
dt.fit(x_train,y_train)
dt.score(x_train,y_train)
pred_decision=dt.predict(x_test)
dts=r2_score(y_test,pred_decision)
print("r2_score:",dts*100)
dtscore = cross_val_score(dt,x_t,y,cv=3)
dtc=dtscore.mean()
print('Cross Val Score:',dtc*100)
```

```
r2_score: 100.0
Cross Val Score: 100.0
```

```
print("Error:")
print("Mean Absolute Error:",round(mean_absolute_error(y_test,pred_decision),2))
print("Mean Square Error:",round(mean_squared_error(y_test,pred_decision),2))
print("Root Mean Sqaured Error:",round(np.sqrt(mean_squared_error(y_test,pred_decision)),2))
```

```
Error:
Mean Absolute Error: 0.0
Mean Square Error: 0.0
Root Mean Sqaured Error: 0.0
```

```
plt.figure(figsize=(8,6))
plt.scatter(x=y_test,y=pred_decision,color='r')
plt.plot(y_test,y_test,color='b')
plt.xlabel('Actual Credit Paid in 5 Days',fontsize=14)
plt.ylabel('Predicted Credit Paid in 5 Days',fontsize=14)
plt.title('Decision Tree Regression',fontsize=18)
plt.show()
```

R2 score and CV Score are 100 , All The Points are interacting with predicted line

Testing

```
import numpy as np
a=np.array(y_test)
a
```

```
array([1, 0, 0, ..., 1, 1, 0], dtype=int64)
```

```
df_com=pd.DataFrame({"Original":a,"Predicted":pred_decision},index=range(len(a)))
df_com
```

	Original	Predicted
0	1	1.0
1	0	0.0
2	0	0.0
3	1	1.0
4	1	1.0
...	...	...
41914	1	1.0
41915	1	1.0
41916	1	1.0
41917	1	1.0
41918	0	0.0

41919 rows × 2 columns

CONCLUSION

- Key Findings and Conclusions of the Study

  Key Finding: Data is having high outliers and data missing in few rows.

  Inference: User who are active and maintaining avg balance for 30days and 90days are more likely to repay the loan.

  Conclusion: Data need to be more specific in type of recharges like talktime, data, top up recharges or SMS Recharge. Specificity of the data is missing. Considered all the balance maintains is for talk time only.

- Learning Outcomes of the Study in respect of Data Science

Challenges: Handling and treating skewed data / outliers.

Visualisation: It help in understanding the frequency of users who are able to manage 30days / 90days monthly balance, active days on network etc

- Limitations of this work and Scope for Future Work

  Data doesn’t specifies weather the recharges or balance maintains is of talktime or data balance. If the data is more specific in this instance it would be easier to target audience more accurately.

  Scope of future work: It would be more helpful if we can identify the exacta expectation of the user, weather the user is looking for talktime / data / SMS. Introduce packs specific loans can also help in increasing the usage of loan for the credible users,